CSCI-599 Machine Learning Theory
Assignment - 3

Sree Priyanka Uppu
(uppu@usc.edu)

1

1.1 Prove that if there exists some $h \in H_k^n$ that has zero error over S(G) then G is k-colorable.

Given h is consistent with S(G). Let $h = \bigcap_{j=1}^{k} h_j$ be an ERM classifier in H_k^n over S. Using the reduction mentioned in the question, we need to prove that given k, graph G is k-colorable. Let us color the vertices of the graph as: $f(v_j) = min\{j : h_j(e_i) = -1\}$

This algorithm, will assign a color to vertex v_i as atleast one $h_j(e_i)$ is -1(from the reduction that, for every vertex v_i, e_i it is assigned a negative label). This implies that every vertex v_i has a color.

We have established that every vertex has a color. Next we need to show that, no 2 adjacent vertices have the same color. We will be using proof by contradiction. Assume that v_a and v_b are 2 adjacent vertices that have the same color. Then, $h_j(e_a) = h_j(e_b) = -1$. And the connecting edge will be assigned with the label: $h_j(\dfrac{e_a + e_b}{2})$.

Using the fact that halfspaces are convex sets and all points on a line joining a and b also lie in a convex set, implies $h_j(\dfrac{e_a + e_b}{2})$=-1 This is contradictory to the definition of S(G) that for every edge (v_i, v_j) construct an instance $(\frac{e_i+e_j}{2})$ with a positive label.

Thus our algorithm ensures that no 2 adjacent vertices have the same color and thus, graph G is k-colorable.

1.2 Prove that if G is k-colorable then there exists some h $\in H_k^n$ that has zero error over S(G).

Given that G is k-colorable, we need to have an assignment consistent with intersection of k-half spaces.

The color assignment is, $f(v_j) = t, 1 \leq t \leq k$.Let there be k hyper planes and $h_t = sgn(w_t + b)$ where $w_{t,i} = -1$ (w_t is a normal vector to the hyper plane h_t) if $f(v_i) = t$ and $w_{t,i} = 0$ If $f(v_i \neq t)$. Given b=0.6 and assuming h as an intersection of hyperplanes $h_1, ...h_k$ fr $f(v_i) = t$ then, $h_t(E_i) = sgn(-1 + 0.6) = -1$.

As $w_{t,i} = -1$ and e_i is a unit vector in the i^{th} dimension, for the assignment of edges no 2 vertices will have the same color. Therefore, $e(v_i, v_j)$ adjacent to each other $f(v_i) = t$ and $f(v_j = t^{'})$ and $t \neq t^{'}$ and we have that $w_{t,i} = 0$ and $w_{t^{'},i} = 0$ which gives us 3 possibilities. Therefore, $h_t(\frac{e_i+e_j}{2}) = sgn(\frac{-1}{2} + 0.6) = 1$, and only will will be -1 and the other vertex will be 0 at one instance of time. Hence, our algorithm is consistent with the definition of S(G).

1.3 Based on the above, prove that for any k $\geq$ 3, the $ERM_{H_k^n}$ problem is NP-hard.

We have shown that the reduction from k-coloring problem to the $ERM_{H_k^n}$. Also, we know that the k-coloring problem is NP-Hard and the reduction is also polynomial in time. Hence, $ERM_{H_k^n}$ is also NP-Hard.

2 AdaBoost: Show that the error of h_t w.r.t the distribution $D^{(t+1)}$ is exactly 1/2. That is, show that for every t $\in$ [T] $\sum_{i=1}^{m} D_i^{(t+1)} 1_{[y_i \neq h_t(x_i)]} = 1/2$

Stating the AdaBoost algorithm:
1. Initialize $D_1(i) = 1/m$ for all i=1,...m and T= number of iterations 2. for t=1 to T do:
(a). Run a weak learner L on D_t to get the hypothesis h_t which has error ϵ_t wrt D_t
(b).Let $\alpha_t = \frac{1}{2} ln(\frac{1-\epsilon_t}{\epsilon_t})$
(c).$D_{t+1}(i) = \frac{D_t(i).exp[-\alpha_t y_i h_t(x_i)]}{Z_t}$, where z is the normalizing factor such that $\sum_{i=1}^{m} D_{t+1}(i) = 1$
3. Find H(x)=sgn(f(x)), where f(x)=$\sum_{t=1}^{T} a_t h_t(x)$

For a weak learner, let us assume that $\exists$ a h $\in$ H whose D_{t+1}< 1/2. We already know that the normalizing factor, $Z_t = 2\sqrt{\epsilon_t(1 - \epsilon_t)}$ and $\alpha_t = \frac{1}{2} ln(\frac{1-\epsilon_t}{\epsilon_t})$, $\epsilon_t = \sum_{i:h_t(x) \neq y_i} D_t(i)$.

Since, initial distribution is uniform, $\sum_{i=1}^{m} D_t(i) = 1$
Solving,

$$\begin{aligned}
\sum_{i=1}^{m} D_i^{(t+1)} 1_{[y_i \neq h_t(x_i)]} &= \sum_{i=1}^{m} \frac{D_t(i) e^{-\alpha y_i h_t(x_i)} 1_{y_i h_t(x_i) < 0}}{Z_t} \\
&= \sum_{y_i h_t(x_i) < 0}^{m} \frac{D_t(i) e^{\alpha_t}}{Z_t} \\
&= \frac{e^{\alpha_t}}{Z_t} \sum_{y_i h_t(x_i) < 0}^{m} D_t(i) \\
&= \frac{\sqrt{\frac{1-\epsilon_t}{\epsilon_t}}}{2\sqrt{\epsilon_t(1-\epsilon_t)}} \epsilon_t \\
&= \frac{1}{2}
\end{aligned} \tag{1}$$

Hence proved, that the AdaBoost algorithm uses the weighting mechanism to "force" the weak learner to focus on the problematic examples in the next iteration.

3 Variant of Statistical Query Model in which the learning algorithm in addition to the Oracle STAT(c,D) is also given access to the unlabeled random draws from the target distribution D.

Theorem 5.3: "Let C be a concept class and let H be a representation class over X. Then if C is efficiently learnable from statistical queries using H, C is efficiently PAC learnable using H in the presence of classification noise" holds in this variant

It implies that the learning algorithm in the noise model has access to D by ignoring the noisy labels returned by $EX_{CN}^{\eta}(c, D)$.

Suppose that SQ algorithm makes M queries to STAT. Use $EX_{CN}^{\eta}(c, D)$ to simulate each STAT query. Take enough examples each time so that the estimate $\hat{p}_c hi$ is off by $>\tau$ with probability $\leq \delta/M$.

We have $EX^{\eta}_{CN}(c, D)$ and must show how it is used efficiently to simulate STAT(c,D). We have χ, τ and we need to estimate $P_\chi = Pr_{x\ D}[\chi(x, c(x)) = 1]$. Dividing the instance space into X_1, X_2 so that we have a part of sample where noise matters (which is X_1) and the other part being where sample does'nt matter(which is X_2).Also, the partitions are disjoint.
$X_1 = x \in X : \chi(x, 0) \neq \chi(x, 1)$ and $X_2 = x \in X : \chi(x, 0) = \chi(x, 1)$
Defining, $p_1 =$ probability that a point x lies in $X_1 . p_2 =$ probability that a point x lies in $X_2 . p_1 + p_2 = 1$
$p_1 = Pr_D[X_1]$ and $p_2 = Pr_D[X_2]$
P^{CN}_χ can be estimated from $EX^{\eta}_{CN} = Pr_{EX^{\eta}_{CN}(c,D)}[\chi = 1]$
$P^1_\chi = Pr_{EX(c,D_1)[\chi=1]}$ and $P^2_\chi = Pr_{EX(c,D_2)[\chi=1]}$.
From the noisy oracle we can calculate the values of P^{CN}_χ and for p_1 since we can ignore the label for points which fall in D_1. It will make use of the unlabeled examples available to us from D. For a large fraction of samples we can calculate p_1 using the Chernoff Bound.

Solving for,

$$\begin{aligned} P^{CN}_\chi[\chi = 1] &= (1 - \eta)P_\chi + \eta[p_1 Pr_{EX(c,D_1)}[\chi = 0] + p_2 Pr_{EX(c,D_2)}[\chi = 1]] \\ &= (1 - \eta)[p_1 P^1_\chi + p_2 P^2_\chi] + \eta[p_1(1 - p^1_\chi) + p_2 P^2_\chi] \\ &= (1 - 2\eta)p_1 P^1_\chi + p_2 P^2_\chi + \eta p_1 \end{aligned} \tag{2}$$

$$P^1_\chi = \frac{P^{CN}_\chi - p_2 P^2_\chi - \eta p_1}{(1 - 2\eta)p_1} \tag{3}$$

But $P_\chi = p_1 P^1_\chi + p_2 P^2_\chi$ Substituting in the above equation we get,

$$\frac{P_\chi - p_2 P^2_\chi}{p_1} = \frac{P^{CN}_\chi - p_2 P^2_\chi - \eta p_1}{(1 - 2\eta)p_1} P_\chi = \frac{1}{1 - 2\eta} P^{CN}_\chi - \frac{\eta p_1}{1 - 2\eta} - \frac{2\eta}{1 - 2\eta} p_2 P^2_\chi \tag{4}$$

We conclude that the we can generate a STAT (c, D) from the noisy oracle $EX^{\eta}_{CN}(c, D)$.In general, any PAC oracle which is using Chernoff bound to calculate the upper limit can be converted to a STAT oracle.

The concept class of axis aligned rectangles in R^n can be efficiently learned in the variant.

We use the variant of the statistical query model in which we are given access to D in addition to STAT(c,D). The algorithm begins by sampling D and using the inputs drawn to partition n-dimensional space. More precisely, for each dimension i, we use the sample to divide the x_i-axis into d/ϵ intervals with the property that the x_i component of a random point from D is approximately equally likely to fall into any of the intervals.

We now estimate the boundary of the target rectangle separately for each dimension using STAT(c, D). Note that if the projection of the target rectangle onto the x_i-axis does not intersect an interval I of that axis, then the conditional probability p_I that the label is positive given that the input has its x_i component in I is 0. On the other hand, if the target's projection onto I is nonzero and there is significant probability that a positive example of the target has its x_i component in I, then p_I must be significantly larger than 0. Thus, our algorithm can start from the left, and moving to the right, place the left x_i-boundary of the hypothesis rectangle at the first interval I such that p_I is significant (at least polynomial in ϵ/n); note that estimating p_I can be done solely with calls to STAT(c,D) once the intervals are defined for each coordinate. The analogous computation is done from the right, and for each dimension. The result is an efficient (polynomial in 1/ϵ and n) algorithm for learning n-dimensional rectangles from statistical queries, immediately implying a noise-tolerant learning algorithm.

4 Show that if there is an efficient algorithm for PAC learning in the presence of classification noise by an algorithm that is given a noise rate upper bound η_0 (1/2 $> \eta_0 \geq \eta \geq 0$) and whose running time depends polynomially on 1/(1 - 2η_0), then there is an an efficient algorithm that is given no information about the noise rate and whose running time depends polynomially on 1/(1 - 2η).

Given: an efficient PAC learnable algorithm, use the Chernoff bounds to get a bound on the number of samples required to assure that the hypothesis is correct with a probability, δ.

We have $0 \leq p \leq 1$, $0 \leq s\ leq 1$ and m>0 also, m is an integer. Applying

Chernoff bounds we get,

$$LE(p, m, p - s) \leq e^{-2s^2 m} \tag{5}$$

$$GE(p, m, p + s) \leq e^{-2s^2 m} \tag{6}$$

For a PAC learning algorithm we know the Chernoff bound to be:

$$LE(p, m, p - s) \leq \delta \tag{7}$$

$$GE(p, m, p + s) \leq \delta \tag{8}$$

We have, $e^{-2s^2 m} \leq \delta$ and solving for m,

$$m \geq \frac{1}{2s^2} ln(\frac{1}{\delta}) \tag{9}$$

Let $\sigma = (x_1, b_1), (x_2, b_2), ..., (x_m, b_m)$ denote a series of samples from a noisy oracle $EX_\eta(c, D)$ and h is any possible hypothesis. Suppose that $F(h_i, \sigma)$ denote the number of indices,j for which h disagrees with (x_j, b_j). Then, the rate at which a hypothesis h_i disagrees with the examples,(x_j, b_j) is equivalent to the noise rate η. Hence, to estimate the noise rate η we use the minimum disagreement over all hypotheses.

We find this estimate, η by taking samples which are sufficient such that the empirical rate of disagreement for each hypothesis is almost equal to the average. Using an iterative search procedure that tries to find a bound η_0, by successively reducing the gap between the actual noise rate, η and $1/2$. Let us begin with an initial guess that η<$1/4$ and $\eta_0 = 1/4$. During the hypothesis testing, if the value of η fails then we increase the value of $\eta = 3/8$ and then by $\eta = 7/16$ and so on. (Every time the halving distance between the previous η value and $1/2$).

Now, we can test the noise rate by:
1. Drawing some samples and estimate the failure probability of each of the hypothesis $h_i \in H$.
2. The smallest empirical failure rate, $\hat{p}_i = F(h_i, \sigma)/m$, (m= the number of samples) is compared to the current value of η_0 and if we get $\hat{p}_i < \eta_0$, we stop the search and output η_0 as the bound.
3. Else, increase η_0 and keep repeatedly increasing the size drawn at each iteration.

Say, a hypothesis h_i disagrees with a fraction p_i of samples in round r, so we get the bounds on the probability that $|\hat{p}_i - p_i| \geq 2^{-(r+2)}$. We substitute $s = 2^{-(r+2)}, m = m_r and \delta = \delta(N2^{r+2})$ in LE, GE in the above equations:

$$LE(p, m_r, p - 2^{-(r+2)}) \leq \frac{1}{2} \frac{\delta/2}{N2^r} \tag{10}$$

$$GE(p, m_r, p + 2^{-(r+2)}) \leq \frac{1}{2}\frac{\delta/2}{N2^r} \tag{11}$$

It follows that,

$$Pr(|\hat{p}_i - p_i| \geq 2^{-(r+2)}) \geq \frac{\delta/2}{N2^r} \tag{12}$$

Summation over N hypotheses and all rounds, for any round r, we have

$$Pr(|\hat{p}_i - p_i| = 2^{-(r+2)}) \leq \delta/2 \tag{13}$$

Thus, with probability >(1-δ/2), there's a possibility of 2 events: 1. Algorithm ends on or before round, $r^{'} = 1 + \lceil \frac{1}{log(1-2\eta)} \rceil$ 2. When the algorithm ends, the empirical estimate of noise, $\hat{\eta_0} > \eta$

In round $r^{'}$, $\eta \leq \frac{1}{2} - \frac{1}{2^{r^{'}}}$ and the number of samples in the round, $m_{r^{'}}$ is sufficient to ensure that $\hat{p}_{min} \leq \eta + 2^{-(r^{'}+2)}$ with probability >1-δ/2. However,

$$\begin{aligned}
\hat{\eta_0} - 2^{-(r^{'}+2)} &= (\frac{1}{2} - \frac{1}{2^{-(r^{'}+2)}}) - \frac{1}{2^{(r^{'}+2)}} \\
&> (\frac{1}{2} - \frac{1}{2^{r^{'}}}) - \frac{1}{2^{(r^{'}+2)}} \\
&\geq \eta + 2^{-(r^{'}+2)} \\
&\geq \hat{p}_{min}
\end{aligned} \tag{14}$$

with probability $> 1 - \delta/2$. Thus, the algorithm will end at or before round $r^{'}$ with the probability.

Suppose the algorithm ends at round $r^{'}$. Then, $\hat{p}_min \geq \eta - 2^{-(r+2)}$ for m_r with probability more than $1 - \delta/2$. Since it ends, $\hat{p}_min \leq \hat{\eta_0} - 2^{-(r+2)}$. Thus,

$$\eta - \frac{1}{2^{r+2}} < \hat{\eta_0} - \frac{1}{2^{r+2}} \tag{15}$$

and hence, $\eta < \hat{\eta_0}$ with probability $> 1 - \delta/2$.

Finally, the algorithm fails when atleast one of the above conditions fails. And each possibility occurs with a probability of atmost $\delta/2$ and failure occurs with a probability of δ which is the confidence parameter from the definition of PAC learning algorithm. Assuming that the algorithm ends at round $r_0 = 1 + \lceil \frac{1}{log(1-2\eta)} \rceil$, the total number of examples is,

$$m_{r_0} = O(\frac{1}{(1-2\eta)^2} ln \lceil \frac{N}{(1-2\eta)\delta} \rceil) \tag{16}$$

Hence proved that can efficiently learn the concept target class in polynomial $(\frac{1}{(1-2\eta)})$.

5 Give an efficient Statistical Query (SQ) algorithm for the class of decision lists. Analyze the complexity of your algorithm.

Let F_n be a boolean valued functions on domain $\{0,1\}^n$. Assume F_n contains the constant function 1. A probabilistic decision list c, over F_n is $(f_1, r_1), (f_2, r_2), ..., (f_s, r_s)$ where each $f_i \in F_n$ and each $r_i \in [0,1]$ and f_s is a constant function 1.

For any $x \in X, c(x) = r_j$ then j is the least index for $f_j(x) = 1$. The functions in F_n are tested one by one in the order of the list specified by F_n, till a f_i evaluates to 1 on which x is encountered which implies, r_j= probability that x=1.

So, c is a probabilistic decision list with ω converging probabilities for $\omega \in [0,1]$ if $|r_i - \omega| \geq |r_{i+1} - \omega| \quad for \quad 1 \leq i < s$.

Case (i): ω =0 , then the learning algorithm learns the decision lists with decreasing probabilities, that is the lists in which $r_i \geq r_j$ for $i \leq j$.

Case (ii): $\omega = 1/2$, then the c is a decision list with decreasing probability as instances with most certain outcomes are handled at the beginning of the list, F_n.

Algorithm :

Input: $\omega \in [0,1]$, basis $F_n = f_1, f_2, ..., f_s$ and $\epsilon, \delta, \gamma > 0$ access to random examples of the decision lists, $F_n with \quad \omega$ converging probability.

Output: Learning algorithm with probability of atleast $1 - \delta$ and (ϵ, γ) model of probability.

Draw a large sample,S of size m polynomial in $(1/\epsilon, s, log 1/\delta, 1/\gamma)$of random examples, where s=$|F_n|$

Using the obtained sample, we define p_i= probability that a random positive example (x,1) is drawn given $f_i(x) = 1$. From, the ω converging decision lists, we know that $|p_1 - \omega| \geq |p_i - \omega|$ for all i. This definition helps us in identifying the first variable of the list, if our estimates $\hat{p}_i$ are sufficiently accurate, we would expect $|\hat{p}_i - \omega|$to be maximized when i= 1.the function f_i for which $|\hat{p}_i - \omega|$ is greatest is placed at the head of the hypothesis list. The remainder of the list is constructed iteratively using the part of the sample on which $f_i(x) = 0$.

Iterate through the list of s examples, if the count of $(x, b) \in S$ and $f_j(x) = 1$ is $\leq m\epsilon/4s$ then assign that j value to p and initialize $\hat{p}_t$ to 0. Else, for the $j \in J$, $\hat{p}_j$ is the fraction of sum of all $f_j(x) = 1$ and b=1 by the count of all where $f_j(x) = 1$ and choose t to be the maximum value of all $|\hat{p}_j - \omega|$

At each iteration, append L with $(f_t, \hat{p}_t)$ and finally output L.

Analysis :

For $I \subset \{1, 2, ..., s\}$ and j $\in \{1, 2, ..., s\}$ let A(I,j) be the set of all instances x for which $f_j(x) = 1$ and $f_i(x) = 0$ for all i∈I.Let

$u(I, j) = Pr_{x\in D}[x \in A(I, j)]$ and $v(I, j) = Pr_{(x,b)\in EX}[b = 1|x \in A(I, j)]$ And let, $\hat{u}(I, j)$ and $\hat{v}(I, j)$ be the corresponding empirical estimates of these quantities.

From multiplicative form of Chernoff bounds, $u(I, j) > \epsilon/2s$

$$\hat{u}(I, j) \geq 1/2.u(I, j) \tag{17}$$

with probability 1-$\delta/(2^{s+1}s)$

Further, if $\hat{u}(I, j) > \epsilon/4s$ then the number of instances in S is atleast, $m\epsilon/4s \geq (8\epsilon^2\gamma^2).ln(2^{s+2}s/\delta)$, by applying additive form of Chernoff bounds we have,

$$|v(I, j) - \hat{v}(I, j)| \leq \epsilon\gamma/4 \tag{18}$$

with probability 1-$\delta/(2^{s+1}s)$, assuming $\hat{u}(I, j) > \epsilon/4s$.
Thus, with probability 1-δ, a sample S is chosen we have,

$$u(I, j) \leq max(\epsilon/2s, 2\hat{u}(I, j)) \tag{19}$$

and whenever we have $\hat{u}(I, j) > \epsilon/4s$ we have,

$$|v(I, j) - \hat{v}(I, j)| < \epsilon\gamma/4 \tag{20}$$

Similarly assume, the empirical estimates $\hat{u}(I, j)$ and $\hat{v}(I, j)$ satisfy the above conditions. Now, we need to prove that this assumption implies that the algorithm's output hypothesis h is an (ϵ, γ)-good model of probability.
Suppose h is given by the list $(f_{t_1}, r_1^{'}), ..., (f_{t_s}, r_s^{'})$. Let $T_i = t_1, ...t_i$. To prove that h is an (ϵ, γ) good model of probability, we show that for $1 \leq i \leq s$, either

$$Pr_{x\in D}[x \in A(T_{i-1}, t_i)] \leq \epsilon/2s \tag{21}$$

or

$$Pr_{x\in D}[|h(x) - c(x)| > \gamma|x \in A(T_{i-1}, t_i)] \leq \epsilon/2 \tag{22}$$

$A(T_{i-1}, t_i)$ are disjoint which implies,

$$\begin{aligned} &Pr_{x\in D}[|h(x) - c(x)| > \gamma] \\ &= \sum_{i=1}^{s} Pr_{x\in D}[|h(x) - c(x)| > \gamma|x \in A(T_{i-1}, t_i)].Pr_{x\in D}[x \in A(T_{i-1}, t_i)] \\ &\leq \epsilon \end{aligned} \tag{23}$$

For the i^{th} iteration of our algorithm, the hypothesis list $(f_{t_1}, r_1^{'}), ..., (f_{t_{i-1}}, r_{i-1}^{'})$. Let $C_j = A(T_{i-1}, j)$ and $p_j = v(T_{j-1}, j)$ and we observe that $\hat{p}_j = \hat{v}(T_{j-1}, j)$ (Since, all examples(x,b) in S are such that $f_k(x) = 0$ for k$\in T_{i-1}$).

If t was chosen, then $\hat{u}(T_{i-1}, t) < \epsilon/4s$, and so $\hat{u}(T_{i-1}, t) < \epsilon/2s$.Thus, $Pr_{x\in D}[x \in A(T_{i-1}, t_i)] \leq \epsilon/2s$ holds.

Else, that is if t was not chose, for all $j \in J, \hat{u}(T_{i-1}, j) > \epsilon/4s$ and thus, $|p_j - \hat{p}_j| \leq \epsilon/4$ we need to prove that, $Pr_{x\in D}[|\hat{p}_t - c(x)| > \gamma|x \in C_t] \leq \epsilon/2$.

Let u be the smallest member of J. Then $p_u = r_u$, by the definition of decision lists. Since, c is given by a list of ω converging probabilities, $|r_u - \omega| \geq |r_j - \omega|$ for $j \geq u$.Thus by our choice of t,

$$|r_j - \omega| \leq |r_u - \omega| = |p_u - \omega| \leq |\hat{p}_u - \omega| + \epsilon\gamma/4 = |\hat{p}_t - \omega| + \epsilon\gamma/4 \tag{24}$$

Suppose, $\hat{p}_t \geq \omega$ then, $r_j \leq \hat{p}_t + \epsilon\gamma/4 for j \in J$ and thus, $c(x) \leq \hat{p}_t + \epsilon\gamma/4 \leq \hat{p}_t + \gamma$ when $x \in C_t$. Let z be the probability that an x is chosen for which c(x) $<\hat{p}_t - \gamma$, given that $x \in C_t$:

$$z = Pr_{x\in D}[c(x) < \hat{p}_t - \gamma|x \in C_t] \tag{25}$$

$$\begin{aligned}
p_t &= E_{x\in D}[c(x)|x \in C_t] \\
&\leq z(\hat{p}_t - \gamma) + (1 - z)(\hat{p} + \epsilon\gamma/4) \\
&\leq z(p_t + \epsilon\gamma/4 - \gamma) + (1 - z)(p_t + \epsilon\gamma/2) \\
&\leq p_t + \epsilon\gamma/2 - \gamma z
\end{aligned} \tag{26}$$

This implies that, $z \leq \epsilon/2$ hence, $Pr_{x\in D}[|\hat{p}_t - c(x)| > \gamma|x \in C_t] \leq \epsilon/2$ holds.

Hence, stating the theorem, let $\omega \in [0, 1]$ be fixed and let F_n be the basis of functions. Then, the p-concept class of probabilistic decision lists over basis F_n with ω - converging probabilities is learnable with a model of probability. (assuming both ω and F_n are known). Specifically, this class can be learned in time polynomial in $(1/\epsilon, 1/\gamma, log(1/\delta), n, |F_n|$ and the maximum time needed to evaluate any function). Thus, also proved that the algorithm of clearly runs in polynomial time. The algorithm is learnable when noisy as well.