Dever de casa # 1

Entrega: segunda-feira, 05/10/2015 (23:59)
Atenção: Justifique todas suas respostas.

1. (**Problem 1.3**)Prove that the PLA eventually converges to a linear separator for separable data. The following steps will guide you through the proof. Let w^* be an optimal set of weights (one which separates the data). The essencial idea in this proof is to show that PLA weights *w(t)* get "more aligned" with w^* with every iteration. For simplicity, assume that $w(0) = 0$.

   (a) Let $\rho = \min_{1 \le n \le N} y_n(w^{*\mathrm{T}} x_n)$. Show that $\rho > 0$.

   (b) Show that $w^{\mathrm{T}}(t)w^* \ge w^{\mathrm{T}}(t-1)w^* + \rho$, and conclude that $w^{\mathrm{T}}(t)w^* \ge t\rho$.
   [Hint: Use induction.]

   (c) Show that $\|w(t)\|^2 \le \|w(t-1)\|^2 + \|x(t-1)\|^2$.
   [Hint: $y(t-1) \cdot (w^{\mathrm{T}}(t-1)x(t-1)) \le 0$ because $x(t-1)$ was misclassified by $w(t-1)$.]

   (d) Show by induction that $\|w(t)\|^2 \le tR^2$, where $R = \max_{1 \le n \le N} \|x_n\|$.

   (e) Using (b) and (d) show that

   $$\frac{w^{\mathrm{T}}(t)}{\|w(t)\|} w^* \ge \sqrt{t}\frac{\rho}{R},$$

   and hence prove that

   $$t \le \frac{R^2 \|w^*\|^2}{\rho^2}.$$

   [Hint: $\dfrac{w^{\mathrm{T}}(t)w^*}{\|w(t)\| \, \|w^*\|} < 1$. *Why?]*

2. (**Problem 1.10**)Assume that $\mathcal{X} = \{x_1, x_2, ..., x_N, x_{N+1}, ..., x_{N+M}\}$ and $\mathcal{Y} = \{-1, +1\}$ with an unknown target function $f : \mathcal{X} \to \mathcal{Y}$. The training data set $\mathcal{D}$ is $(x_1, y_1), ...(x_N, y_N)$. Define the *off-training-set error* of a hypothesis h with respect to f by

   $$E_{\text{off}}(h, f) = \frac{1}{M} \sum_{m=1}^{M} [h(x_{N+m}) \ne f(x_{N+m})].$$

   (a) Say $f(x) = +1$ for all x and

   $$h(x) = \begin{cases} +1 & \text{for all } x = x_k \text{ and k is odd } 1 \le k \le M+N \\ -1 & \text{otherwise} \end{cases}$$

   What is $E_{\text{off}}(h, f)$?

(b) We say that a target function f can 'generate' $\mathcal{D}$ in a noiseless setting if $y_n = f(x_n)$ for all $(x_n, y_n) \in \mathcal{D}$. For a fixed $\mathcal{D}$ of size N, how many possible $f : \mathcal{X} \to \mathcal{Y}$ can generate $\mathcal{D}$ in a noiseless setting?

(c) For a given hypothesis h, and an integer k between 0 and M. how many of those f in (b) satisfiy $E_{\text{off}}(h, f) = \dfrac{k}{M}$?

(d) For a given hypothesis h, if all those f that generate $\mathcal{D}$ in a noiseless setting are equally likely in probability, what is the expected off-training-set error $\mathbb{E}_f[E_{\text{off}}(h, f)]$?

(e) A deterministic algorithm A is defined as a procedure that takes $\mathcal{D}$ as input, and outputs a hypothesis $h = A(\mathcal{D})$. Argue that for any two deterministic algorithms A_1 and A_2,

$$\mathbb{E}_f[E_{\text{off}}(A_1(\mathcal{D}, f))] = \mathbb{E}_f[E_{\text{off}}(A_2(\mathcal{D}, f))].$$

You have now proved that in a noiseless setting, for a fixed $\mathcal{D}$, if all possible f are equally likely, any two deterministic algorithms are equivalent in terms of the expected off-training-set error. Similar results can be proved for more general settings.

3. (**Problem 1.12**)This problem investigates how changing the error measure can change the result of the learning process. You have N data points $y_1 \leq \ldots \leq y_N$ and wish to estimate a 'representative' value.

(a) If your algorithm is to find the hypothesis h that minimizes the in-sample sum of squared deviations.

$$E_{\text{in}}(h) = \sum_{n=1}^{N} (h - y_n)^2,$$

then show that your estimate will be the in-sample mean,

$$h_{\text{mean}} = \frac{1}{N} \sum_{n=1}^{N} y_n.$$

(b) If your algorithm is to find the hypothesis h that minimizes the in-sample sum of absolute deviations,

$$E_{\text{in}}(h) = \sum_{n=1}^{N} |h - y_n|,$$

then show that your estimate will be the in-sample median h_{med}, which is any value for which half the data points are at most h_{med} and at half of the data points are at least h_{med}.

(c) Suppose y_N is perturbed to $y_N + \epsilon$, where $\epsilon \to \infty$. So, the single data point y_N becomes an outlier. What happens to your two estimators h_{mean} and h_{med}?

4. In this problem, you will create your own target function f and dataset $\mathcal{D}$ to see how the Perceptron Learning Algorithm works. Take $d = 2$ so you can visualize the problem, and assume $\mathcal{X} = [-1, 1] \times [-1, 1]$ with uniform probability of picking each $x \in \mathcal{X}$.
   In each run, choose a random line in the plane as your target function f (do this by taking two random, uniformly distributed points in $[-1, 1] \times [-1, 1]$ and taking the line passing through them), where one side of the line maps to $+1$ and the other maps to -1. Choose the inputs x n of the data set as random points (uniformly in $\mathcal{X}$), and evaluate the target function on each x_n to get the corresponding output y_n.

   Now, in each run, use the Perceptron Learning Algorithm to find g. Start the PLA with the weight vector w being all zeros (consider $\text{sign}(0) = 0$, so all points are initially misclassified), and at each iteration have the algorithm choose a point randomly from the set of misclassified points. We are interested in two quantities: the number of iterations that PLA takes to converge to g, and the disagreement between f and g which is $\mathbb{P}[f(x) = g(x)]$ (the probability that f and g will disagree on their classification of a random point). You can either calculate this probability exactly, or approximate it by generating a sufficiently large, separate set of points to estimate it.

   In order to get a reliable estimate for these two quantities, you should repeat the experiment for 1000 runs (each run as specified above) and take the average over these runs.

   (a) Take $N = 10$. How many iterations does it take on average for the PLA to converge for N = 10 training points?

   (b) What is the value of $\mathbb{P}[f(x) = g(x)]$ for $N = 10$?

   (c) Now, try $N = 100$. How many iterations does it take on average for the PLA to converge for N = 100 training points?

   (d) What is the value of $\mathbb{P}[f(x) = g(x)]$ for $N = 100$?

5. (**Problem 1.5**) The perceptron learning algorithm works like this. In each iteration t, pick a random $(x(t), y(t))$ and compute the 'signal' $s(t) = w^T(t)x(t)$. If $y(t) \cdot s(t) \leq 0$, update w by

   $$w(t + 1) \longleftarrow w(t) + y(t) \cdot x(t);$$

   One may argue that this algorithm does not take the 'closeness' between $s(t)$ and $y(t)$ into consideration. Let's look at another perceptron learning algorithm: In each iteration, pick a random $(x(t), y(t))$ and compute $s(t)$. If $y(t) \cdot s(t) \leq 1$, update w by

   $$w(t + 1) \longleftarrow w(t) + \alpha \cdot (y(t) - s(t)) \cdot x(t),$$

   where α is a constant. That is, if $s(t)$ agrees with $y(t)$ well (their product is ¿ 1), the algorithm does nothing. On the other hand, if $s(t)$ is further from $y(t)$, the algorithm changes $w(t)$ more. In this problem, you are asked to implement this algorithm and study its performance.

(a) Generate a training data set of size 100 similar to that used in exercise 4. Generate a test data set of size 10000 from the same process. To get g, run the algorithm above with $\alpha = 100$ on the training data set, until a maximum of 1000 updates has been reached. Plot the training data set, the target function f, and the final hypothesis g on the same figure. Report the error on the test set.

(b) Use the data set in (a) and redo everything with $\alpha = 1$.

(c) Use the data set in (a) and redo everything with $\alpha = 0.01$.

(d) Use the data set in (a) and redo everything with $\alpha = 0.0001$.

(e) Compare the results that you get from (a) to (d).

The algorithm above is a variant of the so-called Adaline (*Adaptative Linear Neuron*) algorithm for perceptron learning.