CS 5350/6350: Machine Learining Spring 2015

Homework 2 Solution

Handed out: Feb 4, 2015
Due date: Feb 18, 2015

1 Boolean Functions

1. There are many possible boolean functions for this problem. Solution that produces right label (y) is acceptable.

   (a) $y = x_1 \wedge x_3$ [1 point].

   (b) $y = x_1 \wedge x_4$ [1 point].

   (c) $y = x_3 \vee x_4$ [1 point].

2. Similar to the precious question, write down the right number of mistakes will get full grades. [5 points]

3. $x_3 + x_4 >= 2$ or simply $x4 >= 1$ [7 points].

2 Mistake Bound Model of Learning

1. Since r is an integer and $1 \leq r \leq 128$, the size of concept class $\mathcal{C}$ is 128 [5 points].

2. If 1) $y^t = 1$ and $(x_1^t)^2 + (x_2^t)^2 > r^2$ or 2) $y^t = -1$ and $(x_1^t)^2 + (x_2^t)^2 \leq r^2$, then current hypothesis has made a mistake [5 points].

3. Since r is an integer, if $y^t = 1$ and $(x_1^t)^2 + (x_2^t)^2 > r^2$, then update r to

$$\left\lceil \sqrt{(x_1^t)^2 + (x_2^t)^2} \right\rceil$$

   If $y^t = -1$ and $(x_1^t)^2 + (x_2^t)^2 \leq r^2$, then update r to

$$\left\lceil \sqrt{(x_1^t)^2 + (x_2^t)^2} \right\rceil - 1$$

   [10 points].

4. The algorithms

Algorithm 1 Mistake-driven learning algorithm

procedure LEARN-R($\mathbf{x}[\,], y[\,]$)
  $r \leftarrow (1+128)/2 = 64$ ▷ initialize r to 64 so that r converge faster
  for each $\mathbf{x}$ and label y **do**
    if $y^t = 1$ and $(x_1^t)^2 + (x_2^t)^2 > r^2$ **then**
      $r \leftarrow \left\lceil \sqrt{(x_1^t)^2 + (x_2^t)^2} \right\rceil$
    else if $y^t = -1$ and $(x_1^t)^2 + (x_2^t)^2 \leq r^2$ **then**
      $r \leftarrow \left\lceil \sqrt{(x_1^t)^2 + (x_2^t)^2} \right\rceil - 1$
    end if
  end for
  return r
end procedure

Clearly, the maximum number of mistakes of the above algorithms is 64. However, the answer of this problem is not unique. It depends on how we initialize the value of r. If we initialize r to be 1 or 128, then the maximum number of mistakes would be 127 [20 points].

5. (**For 6350 students**)[15 points total]

   (a) initialize $r\_min = 1, r\_max = 128$. In each round, update min_r or r_max so that all values of r in range $[r\_min, r\_max]$ don't make mistake. Do such update until $r\_min = r\_max$.

   (b) Similar to problem 3, but this time we need to check both r_min and r_max.

   (c) Halving algorithm:

Algorithm 2 Halving algorithm

procedure LEARN-R($\mathbf{x}[\,], y[\,]$)
  $r\_min \leftarrow 1; r\_max \leftarrow 128$
  while $r\_min \neq r\_max$ **do**
    $r\_mid = (r\_min + r\_max)/2$
    if $y^t = 1$ and $(x_1^t)^2 + (x_2^t)^2 > (r\_mid)^2$ **then**
      $r\_min \leftarrow \left\lceil \sqrt{(x_1^t)^2 + (x_2^t)^2} \right\rceil$
    else if $y^t = -1$ and $(x_1^t)^2 + (x_2^t)^2 \leq (r\_mid)^2$ **then**
      $r\_max \leftarrow \left\lceil \sqrt{(x_1^t)^2 + (x_2^t)^2} \right\rceil - 1$
    end if
  end while
  return r_min
end procedure

Mistake bound: $\log_2 |r| = \log_2 128 = 7$

3 The Perceptron Algorithm and Its Variants

1. [Sanity check, 10 points]

r	$\mathbf{w}$	b	# mistakes
0.3	$[-0.143, -0.269, 0.415, 0.533]$	0.157	3
0.5	$[-0.043, -0.269, 0.115, 0.233]$	0.043	1
0.75	$[-0.293, -1.019, 0.865, 1.733]$	-0.293	5
1	$[0.457, -1.269, 1.115, 1.233]$	-0.543	3
1.0	$[0.257, -0.369, 0.115, 0.333]$	0.257	4

Note: the values would be different since $\mathbf{w}$ and are b are initialized by random numbers.

2. [Online setting, 15 points]

	Perceptron		
r	# update	training accuracy	test accuracy
0.1	377	0.7651	0.8309
0.2	379	0.7639	0.8122
0.5	386	0.7595	0.8015
0.75	375	0.7664	0.8166
1.0	375	0.7657	0.8289

Table 1: Perceptron

r	Margin Perceptron, $\mu = 0.5$		
	# update	training accuracy	test accuracy
0.1	488	0.6960	0.7511
0.2	426	0.7346	0.7735
0.5	411	0.7439	0.8064
0.75	388	0.7583	0.8063
1.0	394	0.7545	0.7896

r	Margin Perceptron, $\mu = 1.0$		
	# update	training accuracy	test accuracy
0.1	532	0.6685	0.7318
0.2	471	0.7065	0.7380
0.5	432	0.7308	0.7727
0.75	424	0.7358	0.7925
1.0	412	0.7433	0.8053

r	Margin Perceptron, $\mu = 3.0$		
	# update	training accuracy	test accuracy
0.1	646	0.5975	0.7016
0.2	573	0.6430	0.7231
0.5	490	0.6947	0.7402
0.75	462	0.7122	0.7598
1.0	453	0.7178	0.7846

r	Margin Perceptron, $\mu = 5.0$		
	# update	training accuracy	test accuracy
0.1	694	0.5676	0.6602
0.2	632	0.6062	0.7160
0.5	541	0.6629	0.7317
0.75	509	0.6829	0.7523
1.0	483	0.6991	0.7399

Table 2: Margin Perceptron

Note: the values would be different since $\mathbf{w}$ and are b are initialized by random numbers.

3. [Using online algorithms in the batch setting, 20 points]

r	Perceptron, epochs = 3			Perceptron, epochs = 5		
	# update	training	test	# update	training	test
0.1	1076	0.7765	0.8285	1681	0.7905	0.7905
0.2	1083	0.7751	0.8270	1710	0.7869	0.8135
0.5	1072	0.7774	0.8157	1692	0.7892	0.8219
0.75	1072	0.7774	0.8229	1692	0.7892	0.8175
1.0	1082	0.7753	0.8079	1692	0.7892	0.8159

Table 3: Without shuffling

r	Margin Perceptron, $\mu = 0.5$, epochs = 3			Margin Perceptron, $\mu = 0.5$, epochs = 5		
	# update	training	test	# update	training	test
0.1	1327	0.7244	0.7615	2098	0.7386	0.7470
0.2	1239	0.7427	0.7658	1947	0.7574	0.7845
0.5	1174	0.7562	0.8005	1806	0.7750	0.7830
0.75	1130	0.7653	0.8079	1802	0.7755	0.7967
1.0	1102	0.7711	0.8258	1764	0.7802	0.8106
	Margin Perceptron, $\mu = 1.0$, epochs = 3			Margin Perceptron, $\mu = 1.0$, epochs = 5		
r	# update	training	test	# update	training	test
0.1	1475	0.6937	0.7321	2282	0.7156	0.7265
0.2	1331	0.7236	0.7638	2100	0.7383	0.7646
0.5	1229	0.7448	0.7987	1932	0.7593	0.7969
0.75	1169	0.7572	0.8071	1821	0.7731	0.7468
1.0	1146	0.7620	0.7910	1821	0.7731	0.8084
	Margin Perceptron, $\mu = 3.0$, epochs = 3			Margin Perceptron, $\mu = 3.0$, epochs = 5		
0.1	1740	0.6386	0.7083	2622	0.6733	0.7055
0.2	1578	0.6723	0.7333	2393	0.7018	0.7358
0.5	1376	0.7142	0.7497	2141	0.7332	0.7493
0.75	1313	0.7273	0.7656	2051	0.7444	0.7515
1,0	1277	0.7348	0.7693	2015	0.7489	0.7819
	Margin Perceptron, $\mu = 5.0$, epochs = 3			Margin Perceptron, $\mu = 5.0$, epochs = 5		
0.1	1899	0.6056	0.6817	2742	0.6583	0.6683
0.2	1714	0.6440	0.7070	2561	0.6809	0.7166
0.5	1484	0.6918	0.7399	2293	0.7143	0.7428
0.75	1399	0.7095	0.7444	2185	0.7277	0.7510
1.0	1338	0.7221	0.7643	2105	0.7377	0.7647

Table 4: Without shuffling

r	Perceptron, epochs = 3			Perceptron, epochs = 5		
	# update	training	test	# update	training	test
0.1	1027	0.7867	0.6500	1624	0.7976	0.8068
0.2	1044	0.7832	0.7104	1585	0.8025	0.8314
0.5	1043	0.7834	0.7350	1640	0.7956	0.8104
0.75	1034	0.7853	0.6465	1648	0.7946	0.8272
1.0	1022	0.7877	0.7006	1617	0.7985	0.7905

Table 5: With shuffling

r	Margin Perceptron, $\mu = 0.5$, epochs = 3			Margin Perceptron, $\mu = 0.5$, epochs = 5		
	# update	training	test	# update	training	test
0.1	1305	0.7290	0.6598	2084	0.7403	0.7313
0.2	1204	0.7499	0.6651	1910	0.7620	0.7568
0.5	1101	0.7713	0.7179	1746	0.7824	0.8155
0.75	1078	0.7761	0.6315	1717	0.7860	0.7977
1.0	1065	0.7788	0.7150	1705	0.7875	0.8120
r	Margin Perceptron, $\mu = 1.0$, epochs = 3			Margin Perceptron, $\mu = 1.0$, epochs = 5		
	# update	training	test	# update	training	test
0.1	1531	0.6820	0.7303	2295	0.7140	0.6682
0.2	1364	0.7167	0.7526	2065	0.7427	0.6805
0.5	1231	0.7443	0.7424	1908	0.7622	0.6912
0.75	1184	0.7541	0.8049	1807	0.7748	0.6735
1.0	1145	0.7622	0.7994	1790	0.7769	0.6871
r	Margin Perceptron, $\mu = 3.0$, epochs = 3			Margin Perceptron, $\mu = 3.0$, epochs = 5		
	# update	training	test	# update	training	test
0.1	1780	0.6303	0.6952	2659	0.6687	0.6459
0.2	1614	0.6648	0.7130	2438	0.6962	0.6637
0.5	1396	0.7101	0.7500	2153	0.7317	0.7013
0.75	1326	0.7246	0.7639	2032	0.7468	0.6946
1.0	1270	0.7362	0.7703	1948	0.7573	0.7682
	Margin Perceptron, $\mu = 5.0$, epochs = 3			Margin Perceptron, $\mu = 5.0$, epochs = 5		
0.1	1907	0.6039	0.6940	2775	0.6542	0.6266
0.2	1743	0.6380	0.6995	2596	0.6765	0.6687
0.5	1520	0.6843	0.7382	2300	0.7134	0.6885
0.75	1436	0.7018	0.7206	2167	0.7300	0.7070
1.0	1370	0.7155	0.7361	2066	0.7426	0.6855

Table 6: With shuffling

4. (**For 6350 Students**)[Aggressive Perceptron with Margin, 10 points]

Experiment setting: # epochs = 3, $r = 0.5\mu = 0, 0.5$(feel free to choose other parameters).

r	Perceptron, $\mu = 0$, no shuffling			Perceptron, $\mu = 0$, shuffling		
	# update	training	test	# update	training	test
0.5	1126	0.7661	0.7855	1190	0.7529	0.8047

r	Margin Perceptron, $\mu = 0.5$, no shuffling			Perceptron, $\mu = 0$, shuffling		
	# update	training	test	# update	training	test
0.5	1875	0.6106	0.6233	1912	0.6029	0.5898