CS 5350/6350: Machine Learning Fall 2021

Homework 2

October 19, 2021

1 Paper Problems [40 points + 8 bonus]

1. [5 points]

   (a) [2 points]

      i. [1 point]
      We prefer L_2's result hypothesis. This is because by Occam's Razor principle we prefer simpler explanations over more complex ones. Thus a smaller hypothesis space means a simpler explanation.

      ii. [1 point]
      This principle can be seen in our PAC guarantee in the $log(|H|)$ part. We know the value $log(|H|)$ will increase as the size of H increases therefore needing a larger m. Thus according to the inequality above it will always be a better decision to choose the hypothesis space that will result in the smallest value of $log(|H|)$ (smallest hypothesis space).

   (b) [3 points]
   We can simply use the inequality above and solve for the smallest value of m given that $err_D(h) < 0.1$ (log base 10)

$$m > \frac{1}{\epsilon}\big(\log(3^{10}) + \log\frac{1}{\delta}\big)$$

$$m > \frac{1}{0.1}\big(\log(3^{10}) + \log\frac{1}{0.05}\big)$$

$$m > 60.722$$

   Thus L_1 must have at least 61 training examples.

2. [5 points]

   Let us start by first separating out the first equation into two cases while remaining equivalent to the first equation.

$$\epsilon_t = \frac{1}{2} - \frac{1}{2}\Bigg(\sum_{y_i = h_t(x_i)}^{m} D_t(i) y_i h_t(x_i) + \sum_{y_i \neq h_t(x_i)}^{m} D_t(i) y_i h_t(x_i)\Bigg)$$

From here because we know that both y_i and $h_t(x_i)$ belong to $\{1, -1\}$ for the first sum where $y_i = h_t(x_i)$ we can see that $y_i h_t(x_i)$ will result in 1 and for the second sum where $y_i \neq h_t(x_i)$ we can see that $y_i h_t(x_i)$ will result in -1. Thus we can rewrite the equation as (multiply both sides by 2 and rewrite parenthesis for simplification)

$$2\epsilon_t = \left(1 - \sum_{y_i = h_t(x_i)}^{m} D_t(i)\right) - \sum_{y_i \neq h_t(x_i)}^{m} D_t(i)$$

From here we can see that we are close to the solution. We know that the weights sum to thus we known that the two sums in the above equation when added together will sum to 1. This gives us that $\left(1 - \sum_{y_i = h_t(x_i)}^{m} D_t(i)\right) = \sum_{y_i \neq h_t(x_i)}^{m} D_t(i)$. Using this we can finally simplify to

$$2\epsilon_t = 2 \sum_{y_i = h_t(x_i)}^{m} D_t(i)$$

$$\epsilon_t = \sum_{y_i = h_t(x_i)}^{m} D_t(i).$$

Thus we have proven the equaility.

3. [20 points]

   (a) [2 point] $f(x_1, x_2, x_3) = x_1 \wedge \neg x_2 \wedge \neg x_3$
   hyperplane: $y = 1$ if $x_1 - x_2 - x_3 \geq 1$
   weight vector = $(1, -1, -1)$
   bias parameter = -1

   (b) [2 point] $f(x_1, x_2, x_3) = \neg x_1 \vee \neg x_2 \vee \neg x_3$
   hyperplane: $y = 1$ if $-x_1 - x_2 - x_3 \geq -2$
   weight vector = $(-1, -1, -1)$
   bias parameter = 2

   (c) [8 points] $f(x_1, x_2, x_3, x_4) = (x_1 \vee x_2) \wedge (x_3 \vee x_4)$
   For this boolean function there is not a linear classifier that represents thus we must find a feature mapping. We will create two new features $x_5 = x_1 * x_2$ and $x_6 = x_3 * x_4$. This is because we want $(x_1 \vee x_2)$ and $(x_3 \vee x_4)$ to evaluate to 1 if either of the values are 1 thus subtracting the product will always result in this behaviour.
   hyperplane: $(x1 + x2 - x5) + (x3 + x4 - x6) \geq 2$
   weight vector = $(1, 1, 1, 1, -1, -1)$
   bias parameter = -2

(d) [8 points] $f(x_1, x_2) = (x_1 \wedge x_2) \vee (\neg x_1 \wedge \neg x_2)$

For this boolean function there is not a linear classifier that represents thus we must find a feature mapping. We will create two new features $x_3 = x_1 * x_2$ and $x_4 = \neg x_1 * \neg x_2$. This is because we want $(x_1 \wedge x_2)$ or $(\neg x_1 \wedge \neg x_2)$ to evaluate to 1 thus either both values mush be 0 or both values must be one. We can see that our feature mapping accomplishes this.

hyperplane: $y = 1$ if $x_3 + x_4 \geq 1$

weight vector = $(0, 0, 1, 1)$

bias parameter = -1

4. [**Bonus**] [8 points]

(a) [2 points] $(\mathbf{x}^\top \mathbf{y})^2$

In order to show this we will use the kernel trick to create a feature mapping. Let us have a polynomial kernel such that $K(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^\top \mathbf{y})^2 = (x_1 y_1 + x_2 y_2)^2$. Next we expand the equation giving us $(x_1 y_1 + x_2 y_2)(x_1 y_1 + x_2 y_2) = x_1^2 y_1^2 + 2 x_1 y_1 x_2 y_2 + x_2^2 y_2^2$. From here we can see that doing this multiplication is the same as the multiplying two vectors $\mathbf{v}_1, \mathbf{v}_2$ where $v_1 = (x_1^2, \sqrt{2} x_1 x_2, x_2^2)$ and $v_2 = (y_1^2, \sqrt{2} y_1 y_2, y_2^2)$ Thus we can simply say that $(\mathbf{x}^\top \mathbf{y})^k = \phi(\mathbf{x})^\top \phi(\mathbf{y})$ where $\phi(\mathbf{x}) = \mathbf{v}_1$ and $\phi(\mathbf{y}) = \mathbf{v}_2$.

(b) [2 points] $(\mathbf{x}^\top \mathbf{y})^3$

$(\mathbf{x}^\top \mathbf{y})^k = \phi(\mathbf{x})^\top \phi(\mathbf{y})$ where $\phi(\mathbf{x}) = \mathbf{v}_1$ and $\phi(\mathbf{y}) = \mathbf{v}_2$ and $v_1 = (x_1^3, \sqrt{3} x_1^2 x_2, \sqrt{3} x_1 x_2^2, x_2^3)$, $v_2 = (y_1^3, \sqrt{3} y_1^2 y_2, \sqrt{3} y_1 y_2^2, y_2^3)$

(c) [4 points] $(\mathbf{x}^\top \mathbf{y})^k$ where k is any positive integer.

In order to show this we will use the kernel trick to create a feature mapping for any positive integer k. Let us have a polynomial kernel such that $K(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^\top \mathbf{y})^k = (x_1 y_1 + x_2 y_2)^k$. Next we expand the equation giving us $(x_1 y_1 + x_2 y_2)_1 (x_1 y_1 + x_2 y_2)_2 ... (x_1 y_1 + x_2 y_2)_k$. From here we can see that doing this multiplication is the same as the multiplying two vectors $\mathbf{v}_1, \mathbf{v}_2$. We can see the values for $\mathbf{v}_1, \mathbf{v}_2$ when $k = 2$ in part a and when $k = 3$ in part b. Thus we can simply say that $(\mathbf{x}^\top \mathbf{y})^k = \phi(\mathbf{x})^\top \phi(\mathbf{y})$ where $\phi(\mathbf{x}) = \mathbf{v}_1$ and $\phi(\mathbf{y}) = \mathbf{v}_2$.

5. [10 points]

(a) [1 point]

$$J(\mathbf{w}, b) = \frac{1}{2} \sum_{i=1}^{m} (y_i - (\mathbf{w}^T \mathbf{x}_i + b))^2$$

(b) [3 points]

First we must take the derivative with respect to w.

$$\frac{dJ}{dw_j} = -\sum_{i=1}^{m} (y_i - (\mathbf{w}^T \mathbf{x}_i + b)) x_{ij}$$

From here we can simply plug in all the rows from the provided table to come to a solution. Thus resulting in

$$\frac{\nabla J}{\nabla \mathbf{w}} = (-24, 12, -48).$$

Now take the derivative with respect to b.

$$\frac{dJ}{db} = -\sum_{i=1}^{m}(y_i - (\mathbf{w}^T\mathbf{x}_i + b))$$

Follow the same process as above resulting in

$$\frac{\nabla J}{\nabla b} = (-12, 12, 12).$$

(c) [3 points] What are the optimal $\mathbf{w}$ and $\mathbf{b}$ that minimize the cost function?

We can do this using the equation $\mathbf{w} = (\mathbf{X}\mathbf{X}^\top)^{-1}\mathbf{X}\mathbf{Y}$ with folding the bias term in this results in $\mathbf{w} = (-1, 1, 1, 1)$. Thus $\mathbf{w} = (1, 1, 1)$ and $b = 1$

(d) [3 points] Now, we want to use stochastic gradient descent to minimize $J(\mathbf{w}, b)$. We initialize $\mathbf{w} = \mathbf{0}$ and $b = 0$. We set the learning rate $r = 0.1$ and sequentially go through the 5 training examples. Please list the stochastic gradient in each step and the updated $\mathbf{w}$ and b.

Stochastic Gradient	b	w[1]	w[2]	w[3]
2.000	0.200	0.000	0.000	0.000
1.800	0.200	0.180	0.000	0.000
1.620	0.200	0.180	0.162	0.000
-1.458	0.200	0.180	0.162	-0.146
2.604	0.460	0.180	0.162	-0.146
2.343	0.460	0.414	0.162	-0.146
2.109	0.460	0.414	0.373	-0.146
1.898	0.460	0.414	0.373	0.044
-0.546	0.406	0.414	0.373	0.044
-0.491	0.406	0.365	0.373	0.044
0.442	0.406	0.365	0.417	0.044
-0.398	0.406	0.365	0.417	0.004
2.803	0.686	0.365	0.417	0.004
2.523	0.686	0.618	0.417	0.004
2.271	0.686	0.618	0.644	0.004
4.087	0.686	0.618	0.644	0.413
-3.823	0.304	0.618	0.644	0.413
-3.441	0.304	0.273	0.644	0.413
-9.291	0.304	0.273	-0.285	0.413
0.310	0.304	0.273	-0.285	0.444

2 Practice [60 points + 10 bonus]

1. [2 Points]

`https://github.com/ericodonoghue/machine_learning_uofu`

2. [36 points]

(a) [8 points]

We can see that by looking at our test data for the decision tree we were able to predict almost as well as Adaboost for depth 3 but for all other depths Adaboost performed better. This is because Adaboost will decrease the bias so the increase isn't huge, but it will still perform better than the single decision tree especially the decision tree at larger depths due to over fitting. The decision tree performs better than Adaboost on the training data, this is due to the single decision tree over fitting the training data while Adaboost does not due that.

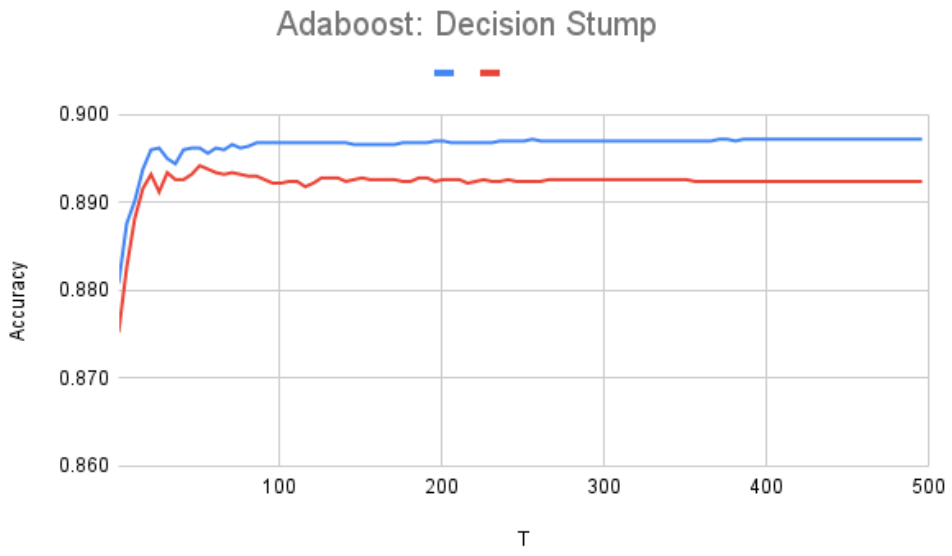

Figure 1: 2.2.a

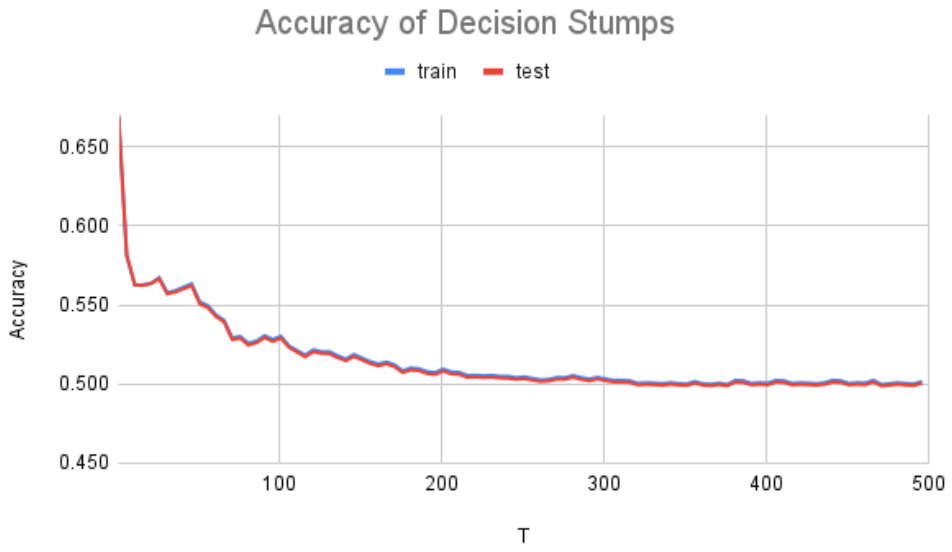

Figure 2: 2.2.a

(b) [8 points]

Yes bagged trees are better than a single tree. This is because averaging all the trees results in a lower variance while a single tree has a much higher variance. Bagged trees seemed to be better than Adaboost first because it ran so much quicker through 500 iterations compared to my Adaboost implementation. Other than that bagged trees helped solve the issue of over fitting the training data while Adaboost decreased the bias. It seems like the better one depends on the situation at hand.

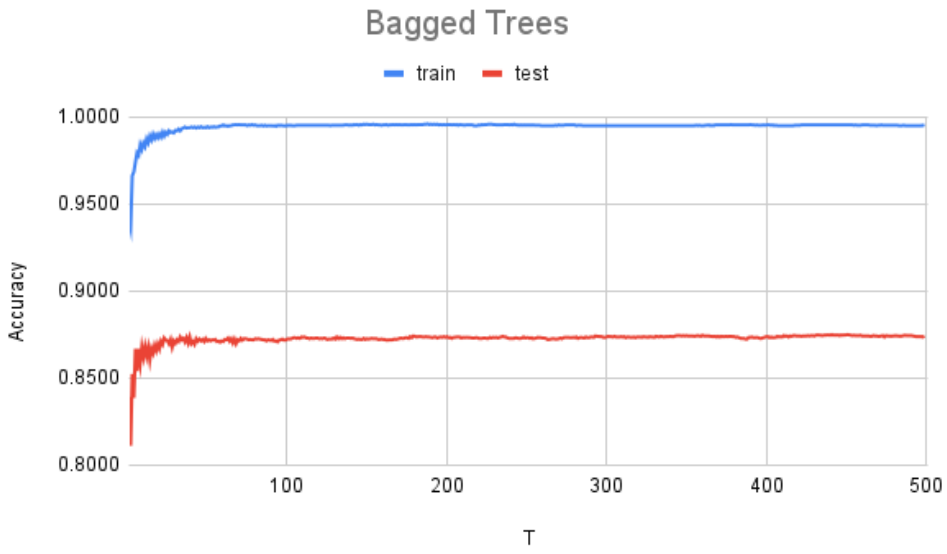

Figure 3: 2.2.b

(c) [6 points]

We can conclude that the bagged trees operate much better than the single tree learners. This is because using the bagged trees reduces the amount of variance while single trees have very high variance. This lower varince in bagged trees results in much better results

Single	Bias	Var	GSE
	4.210	419.852	424.062
Bagged	Bias	Var	GSE
	4.489	4.772	9.261

Figure 4: 2.2.c

(d) [8 points]

The performance of the random forest algorithm seemed a bit better than bagged trees. I believe this is because it only needs to calculate the information gain for a much smaller portion of the columns each time instead of having to do all the columns every time.

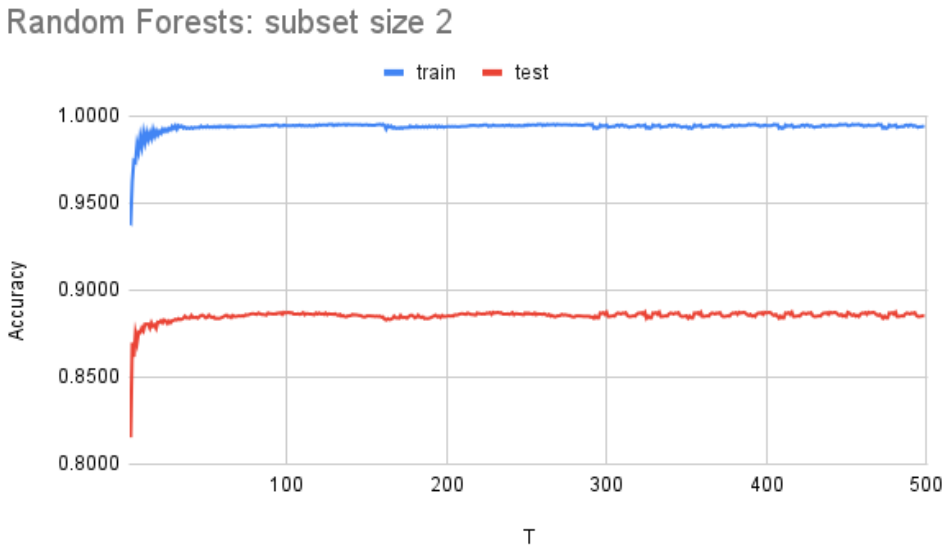

Figure 5: 2.2.d

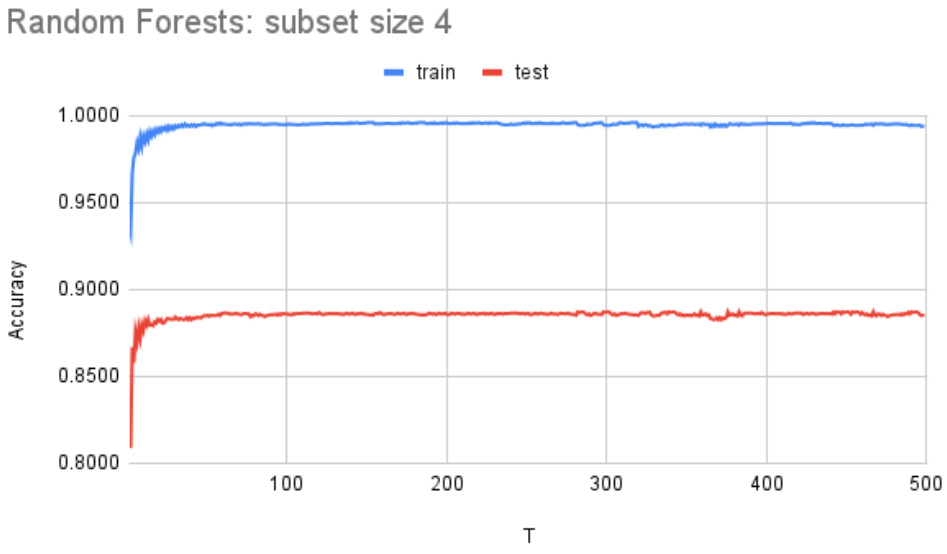

Figure 6: 2.2.d

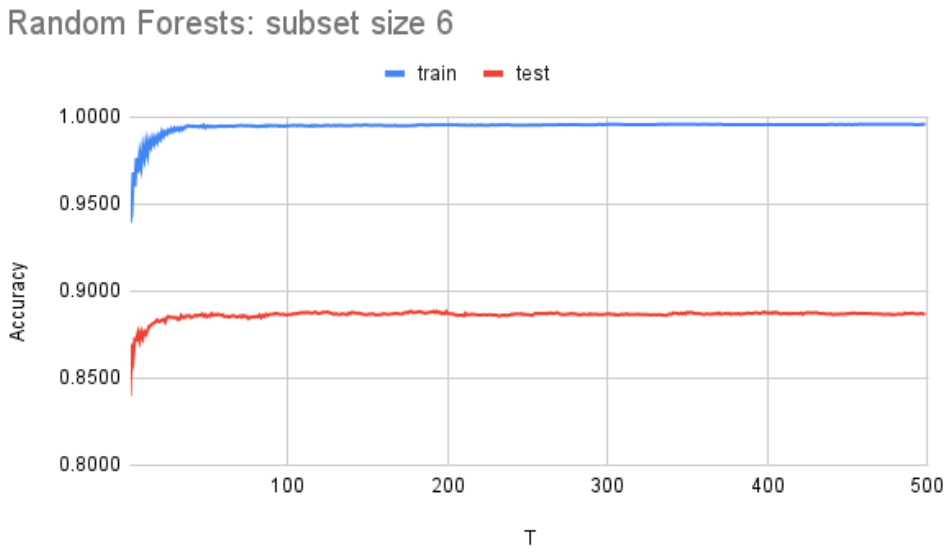

Figure 7: 2.2.d

(e) [6 points]

We can conclude that the random trees are not quite as good as the bagged trees. I believe that this is because the random forest has a slightly larger variance due to only using a smaller set of the attributes to calculate the split. This will result in a slightly higher variance but overall still a very low variance. This shows that due to performance the random forest still has a small enough variance to be used.

Single	Bias	Var	GSE
	4.268	4.311	8.579
Bagged	Bias	Var	GSE
	5.078	5.129	10.207

Figure 8: 2.2.c

3. [**Bonus**][10 points]

4. [22 points]

   (a) [8 points]

Batch Gradient Descent								
Learned Weight Vector:	-0.015	0.900	0.786	0.851	1.299	0.130	1.572	0.998
Learing Rate r:	0.0078125							
Cost of Test Data	23.361							

Figure 9: 2.4.a

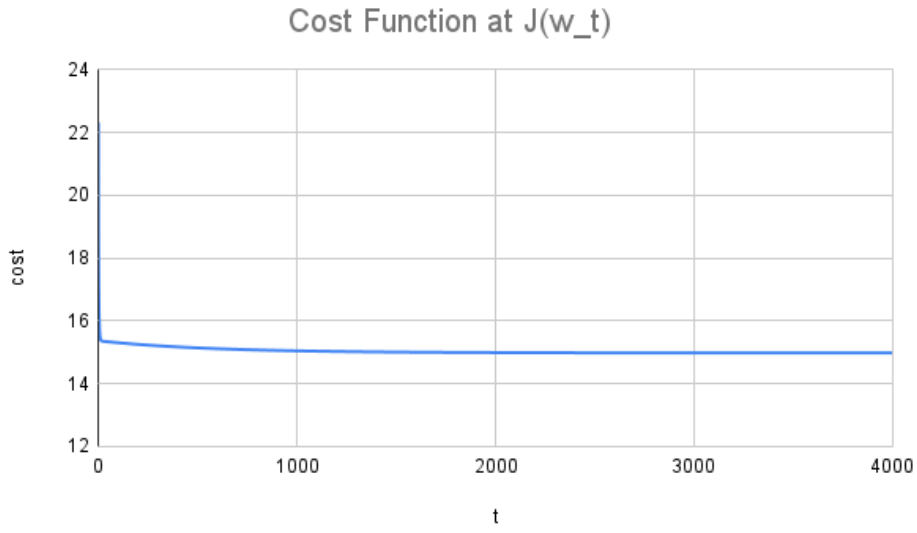

Figure 10: 2.4.a

(b) [8 points]

Stochastic Gradient Descent								
Learned Weight Vector:	-0.054	0.476	0.348	0.388	0.941	0.077	0.991	0.573
Learing Rate r:	0.0078125							
Cost of Test Data:	22.861							

Figure 11: 2.4.b

(c) [6 points]

This weight vector is almost the same as both the vectors learned by batch gradient descent and stochastic gradient descent. This is because the analytical form produces the optimal weight vector but is not viable due to computation power needed. Thus the algorithms we implemented were able to get very close to the optimal vector due to without needing nearly as much computational power. The batch seemed closer, I believe this is because batch is better than stochastic at learning w but is slower.

analytical weight vector:	-0.015	0.901	0.786	0.851	1.299	0.130	1.572	0.999

Figure 12: 2.4.c