Anden aflevering
Data Analysis

Nikolaj Dybdahl Rathcke (rfq695)

May 3, 2015

Question 1

(1)

The perceptron update role on round t is defined by

$$w^T(t+1) = w^T(t) + y(t)x^T(t) \tag{1}$$

We want to prove that

$$y(t)w^T(t)x(t) < y(t)w^T(t+1)x(t)$$

We can use (1) to expand the right hand side, so we get

$$\begin{aligned} y(t)w^T(t)x(t) &< y(t)(w^T(t) + y(t)x^T(t))x(t) \\ &= y(t)w^T(t)x(t) + y(t)y(t)x^T(t)x(t) \end{aligned}$$

subtracting $y(t)w^T(t)x(t)$ from both sides yields

$$\begin{aligned} 0 &< y(t)y(t)x^T(t)x(t) \\ &= x^T(t)x(t) \end{aligned}$$

as $y(t) \in \{-1, 1\}$.
This holds since the RHS will always be positive.

(2)

The files `q1_2a` and `q1_2b` shows how the graphs are generated.
Figure 1 shows a situation where $x(t)$ is misclassified by $w(t)$ but is classified correctly by $w(t+1)$. The green line shows $w(t)$, the blue line shows $w(t+1)$ and the red lines are the orthogonals to each of them. In this situation $w(t) = \begin{pmatrix} 2 \\ 0 \end{pmatrix}$, $x(t) = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$ and $y = -1$. We can then calculate $y(w^T x)$:

$$-1 \cdot \left((2, 0) \begin{pmatrix} 1 \\ 2 \end{pmatrix} \right) = -2$$

As this is less than 0, it is misclassified. After using the first update rule (1), we have that $w(t+1) = \begin{pmatrix} 1 \\ -2 \end{pmatrix}$. Calculating $y(w^T x)$ gives us

$$-1 \cdot \left((1, -2) \begin{pmatrix} 1 \\ 2 \end{pmatrix} \right) = 3$$

Since this is greater than 0 it is classified correctly.

Figure 2 shows a situation where $x(t)$ is misclassified by both $w(t)$ and $w(t+1)$. Again, the green

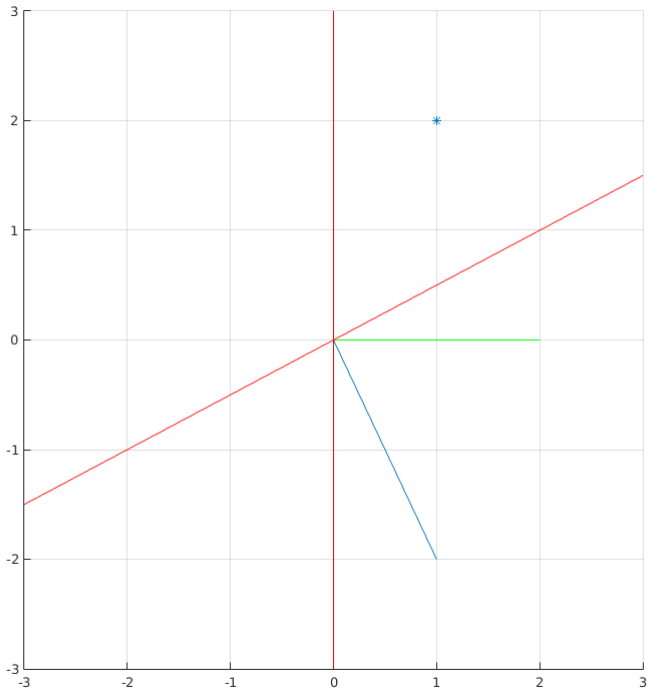

Figure 1: $x(t)$ misclassified by $w(t)$ but classified correctly by $w(t+1)$

line is $w(t)$, the blue is $w(t+1)$ and the red are the orthogonals. In this situation $w(t) = \begin{pmatrix} 2 \\ 2 \end{pmatrix}$, $x(t) = (1, 2)$ and $y = -1$. Calculating $y(w^T x)$ yields:

$$-1 \cdot \left((2, 2) \begin{pmatrix} 1 \\ 2 \end{pmatrix} \right) = -6$$

As this is less than 0, it is misclassified. After using the first update rule, we have that $w(t+1) = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$. Calculating $y(w^T x)$ gives us

$$-1 \cdot \left((1, 0) \begin{pmatrix} 1 \\ 2 \end{pmatrix} \right) = -1$$

This is still less (or equal) than 0, so it is misclassified.

The rotation is sufficient to classify $x(t)$ when the data is separable by a linear combination through the origin. It is not sufficient if data can only be separated if it is required to rotate around another point.

(3)

No. The termination condition is that

$$\exists i \text{ s.t. } y_i(w^T x_i) \leq 0$$

When the data is not separable, it means that there will always exists a data point that satisfies this.

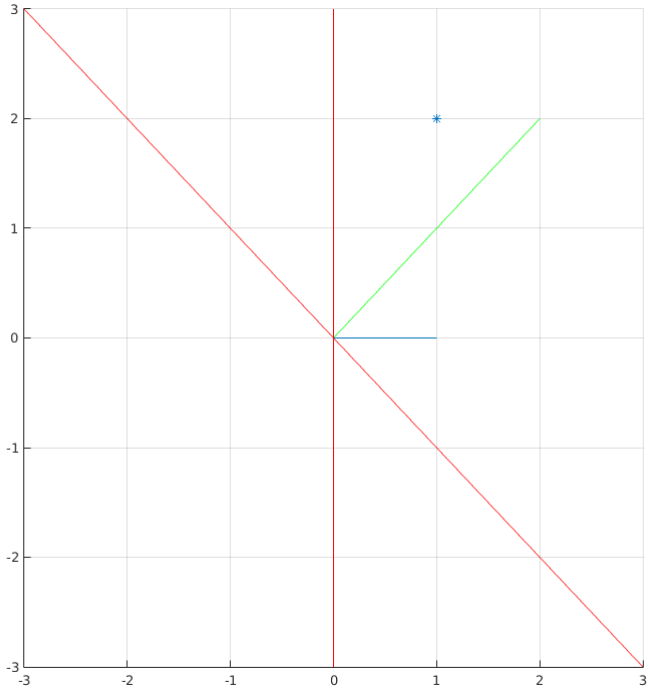

Figure 2: $x(t)$ missclassified by both $w(t)$ and $w(t+1)$

Question 2

(1)

The linear models uses a signal $s = w^T x$ that combines the input variables linearly. The perceptron or pocket algorithm are linear classification models, which uses the signal to produce an output which is ± 1 (the sign function).
The logistic regression model, though its output is still bounded (it takes values between 0 and 1), the produced output is a real number.
In our example, when predicting admission, the linear classification model produces a "yes" or "no" answer, while the logistic regression produces a probability that a student is admitted.

(2)

Running the code to calculate the probability that student 10 is admitted provides the result 0.4731.
In the file `q2_2`, the logistic function Θ is defined and used with the hypothesis h. Running it gives the same result, 0.4731.

(3)

The file `negloglike` implements the function to find the negative log-likelihood, the equation 3.9 [LFD]. Running this from the file `q2_3` gives us the following w and w_est coefficient vectors.

$$w = \begin{pmatrix} -4.9494 \\ 0.7547 \\ 0.2691 \end{pmatrix}$$

$$w_est = \begin{pmatrix} -4.9494 \\ 0.7547 \\ 0.2691 \end{pmatrix}$$

which are the same.

(4)

The function is implemented in the file `errorIn` is based on the equation 3.9 [LFD]. If we use it on w, we get 0.6004.

(5)

The gradient descent function is implemented in the file `gradientDesc`. It takes three parameters. The first one, `init`, is the starting point. The second, `step`, is the step size η and the last one, `t`, is the number of iterations.
It returns two things to make the plot in (7) easy. It returns the last vector w, but also a vector for the values of E_{in} for every 500'th iteration.

(6)

Using the gradient descent function with parameters `init`=$[0, 0, 0]$, `step`=0.1 and `t`=20000 provides the result

$$E_{in} = \begin{pmatrix} -4.8774 \\ 0.7366 \\ 0.2676 \end{pmatrix}$$

The result we got from using `fminunc` was

$$E_{in} = \begin{pmatrix} -4.9494 \\ 0.7547 \\ 0.2691 \end{pmatrix}$$

The parameters has been found experimentally, but the two results are close eachother.

(7)

The relation between iterations and E_{in} is seen in Figure 3 (using 20000 iterations).
The green point is the the value we want to approach. The figure evens out as we get to iteration $t = 10000$ and it's close to the green point, so the choice $t = 20000$ is an accurate estimate.
It has been created with the code from file `generateFigure`.

(8)

We can use the calculation from (2). Changing the vector to the one we found in (6) gives us the probability 0.4886.

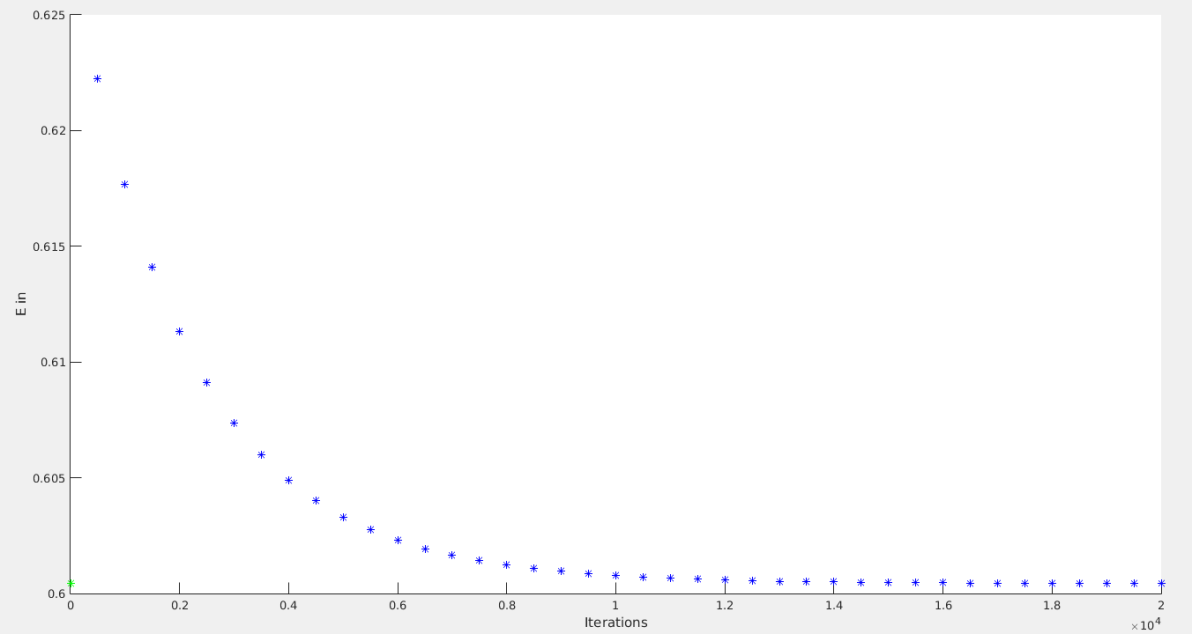

Figure 3: E_{in} as a function of iterations

Question 3

(1)

From equation (2.1) in [LFD] we have that

$$E_{out}(g) \leq E_{in}(g) + \sqrt{\frac{1}{2N} \ln \frac{2M}{\delta}}$$

The generalization bounds for predicting $\mathcal{H}_0$ where the upper bounds for E_{out} holds with probability $1 - \frac{\delta}{2}$ will be (number of hypotheses M is 2).

$$\begin{aligned} E_{out}(g) &\leq E_{in}(g) + \sqrt{\frac{1}{2N} \ln \frac{2 \cdot 2}{\frac{\delta}{2}}} \\ &= E_{in}(g) + \sqrt{\frac{\ln \frac{8}{\delta}}{2N}} \end{aligned}$$

Similarly for $\mathcal{H}_8$, when the number of hypotheses are 2^8 we get

$$\begin{aligned} E_{out}(g) &\leq E_{in}(g) + \sqrt{\frac{1}{2N} \ln \frac{2 \cdot 2^8}{\frac{\delta}{2}}} \\ &= E_{in}(g) + \sqrt{\frac{\ln \frac{2^{10}}{\delta}}{2N}} \end{aligned}$$

Note that the bounds are a bit looser as they use the two-sided Hoeffding bound.