# Week 9 Questions - ST3009: Statistical Methods for Computer Science

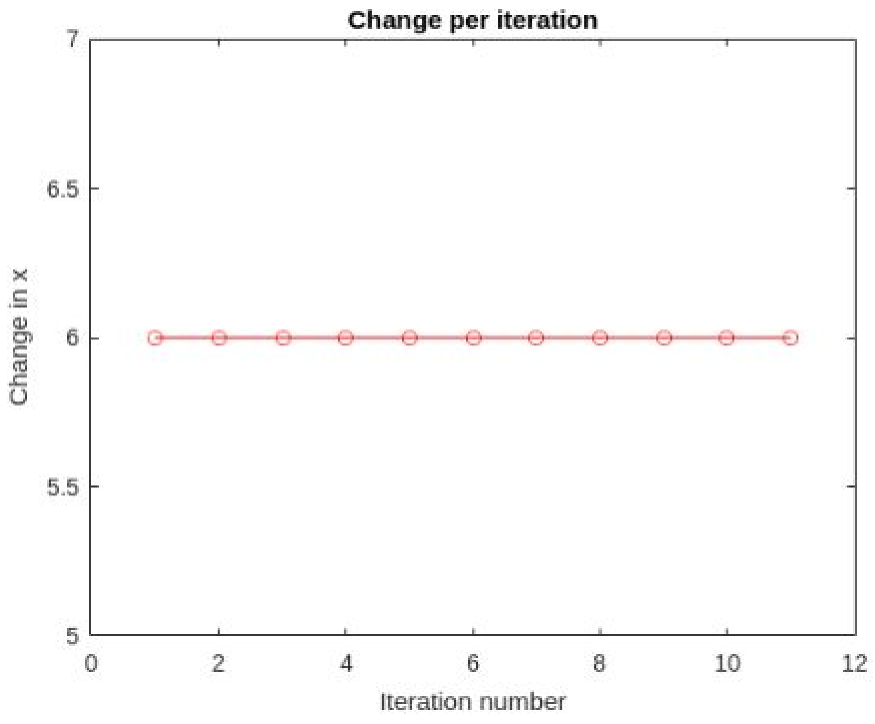
## Samuel Petit - 17333946

### Question 1 a

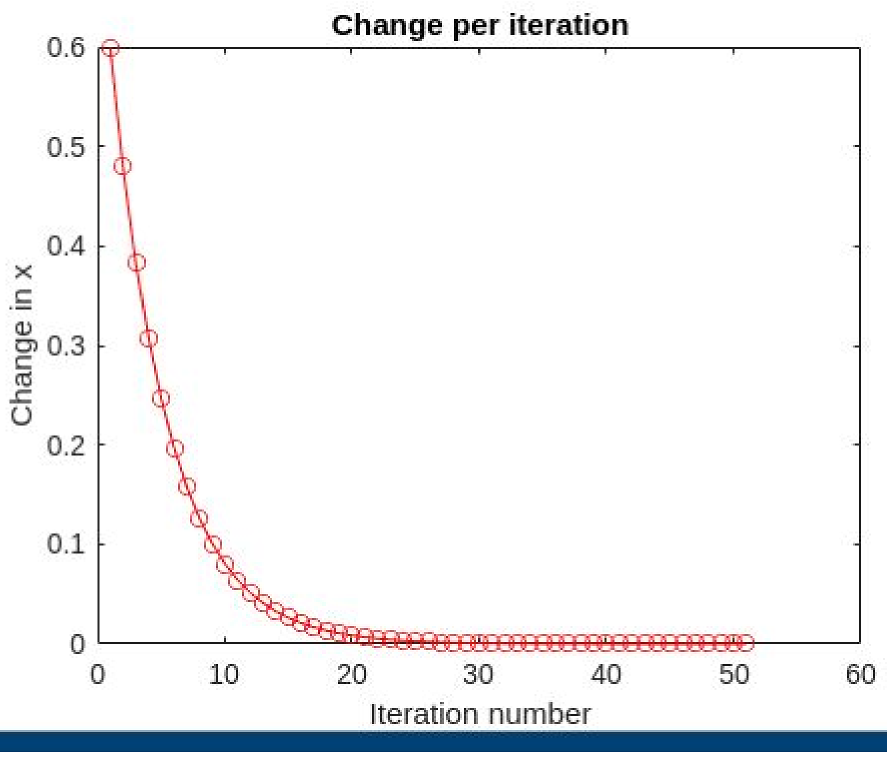
```
x = 3; % starting point for x

% α = 1, α = 0.1 and α = 0.01
alpha = 0.1;
% This value is updated based on alpha so that the graph shows
maxIterations = 100;

% Iterations count
count = 0;
changeAmount = 1;

% Array keeping all changeAmount values for all iterations
changeVals = zeros(1, maxIterations);

% We stop if we reach our maximum iterations amount
while count <= maxIterations
   count = count+1;
   prevX = x;
   x = x - alpha * (2 * x); % compute new x value
   changeAmount = abs(x - prevX);
   changeVals(count) = changeAmount;
end

% Display x estimate and graph of changes
disp(x)
plot(changeVals, '-or') ;
hold on;
title('Change per iteration')
xlabel('Iteration number')
ylabel('Change in x')
```

## Question 1 b

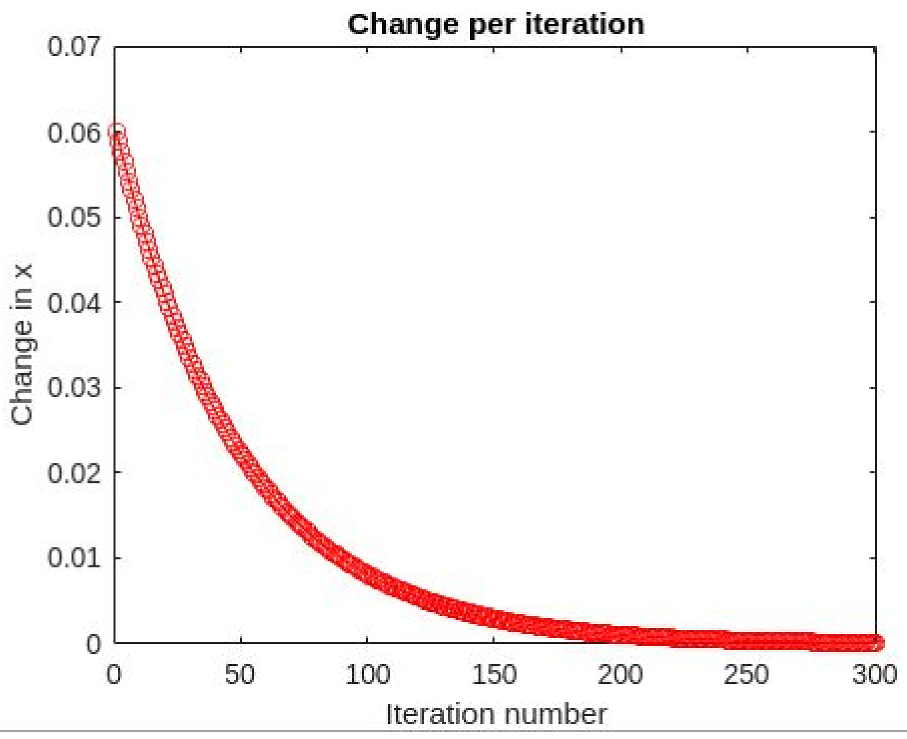
Alpha = 1



Alpha = 0.1



Alpha = 0.01



Observations:

- α = 1 : leaves our x estimate at 3 since in this case it will jump between 3 and -3, Making the change in x at every iteration at 6.
- α = 0.1 Leaves us with an estimate for x of 3.4254e-05, this value is closer to the actual value than when α = 1. We find that the values for x stop varying after about 25 iterations.

- α = 0.01: The amount changed reaches 0 after about 300 iterations, leaving us with an estimate for x of 0.0069, that estimate is also closer than when alpha was 0.1.

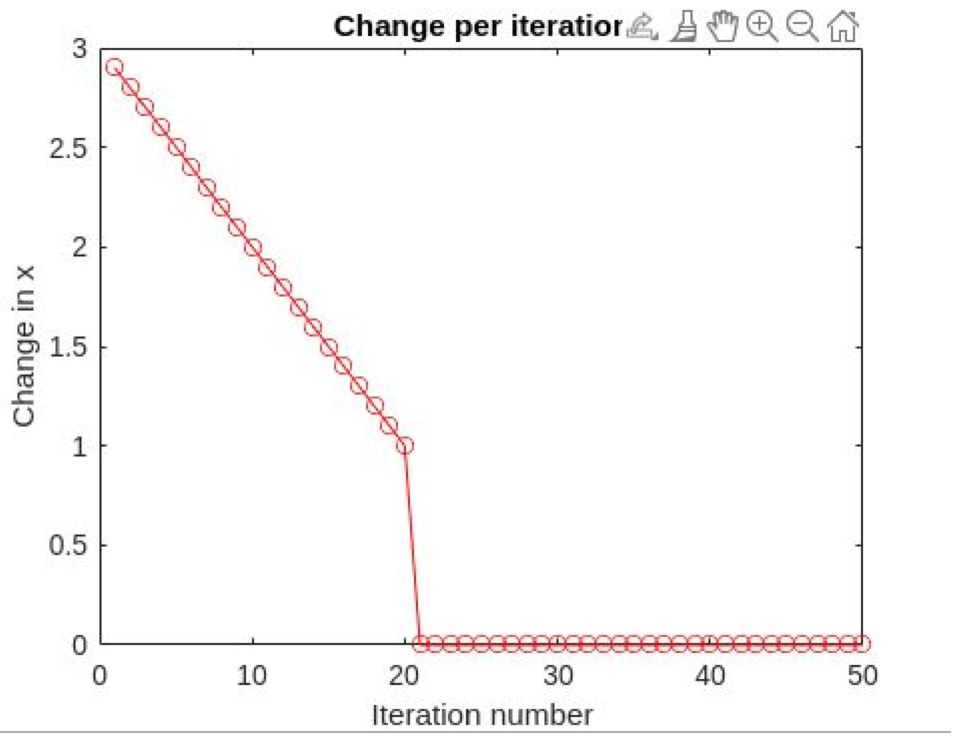
## Question 1 c

Code for this new approach:

```
x = 3; % starting point for x
% α = 1, α = 0.1 and α = 0.01
alpha = 0.1;
% Iterations count
count = 0;
changeAmount = 1;
compute = true;
% Array keeping all changeAmount values for all iterations
changeVals = zeros(1, maxIterations);

% We also stop if we reach our maximum iterations amount
% (could be stuck in an infinite loop otherwise)
while compute
   count = count+1;
   y_prev_x = x * x - 1;
   tmp_x = x - alpha;
   y_next_x = tmp_x * tmp_x - 1;

   if abs(y_prev_x) < abs(y_next_x)
       compute = false;
   else
       x = tmp_x;
       changeAmount = abs(x - prevX);
       changeVals(count) = changeAmount;
   end
end

% Display x estimate and graph of changes
disp(x)
plot(changeVals, '-or') ;
hold on;
title('Change per iteration')
xlabel('Iteration number')
ylabel('Change in x')
```

## Question 1 d

Graph for this new approach:



The way I implemented this new approach is to use an alpha value to increment (or decrement) the current value of x. If the y value obtained with this new x value is closer to 0 than that of the current then we continue the algorithm, otherwise we have reached the minimum value possible for that value of alpha (in this case alpha represents the precision).

An alpha value of 0.1 here seems to give us the correct answer after 20 iterations so it seems that it is generally more accurate than the gradient descent method.

Reviews : positive or negative
Given : set of reviews with outputs (positive / negative)
→ used to build a dictionary of n reviews
For each review : have the word & count of times it was used in the review

## Question 2 a

First of all given how our data was preprocessed we would be using the “Bag of Words” approach.
The next step is to map each remaining word into a feature vector with integer values such that we can apply our functions on the data.

Now we can apply our logistic regression methods as usual with the following loss function :

$\frac{1}{m}\sum_{i=1}^{m}\log(1+e^{-y^{(i)}\theta^T x^{(i)}})$ such as to train our model.

## Question 2 b

We have a couple of assumptions, the first one being that the data is linearly separable. We also assume that we can separate a review from good to bad only based on text and no other factors such as an amount of stars or other parameters. We also assume that the data provided is valid, for instance typos in a review could cause the model to be trained incorrectly, reviews with empty texts could also mess with the training.
Finally, we assume that the words provided in the feature vector are relevant and that irrelevant words have been remove (such as small words like the, of, or…), and similar words such as likable, liking… have been truncated to “lik”, this last part is not essential but would improve the quality of the model after training.

## Question 2 c

Discuss how you could use bootstrapping to estimate a confidence interval for the prediction of the logistic regression classifier for a specified review. Hint: the randomness in the prediction is due to the training data, so consider resampling the training data.

By bootstrapping we can estimate a confidence for the model predictions, this will tell us how sensitive the model is.

To do so we will repeat the following steps:

- Sample with replacement from the training data
- Train the model based on the sample

By doing so we can find a set of parameters which minimise the cost function for this data. We can then use those to estimate the distribution of our parameter values by building a confidence interval.