Linear Regression

1. Linear Regression with One Variable

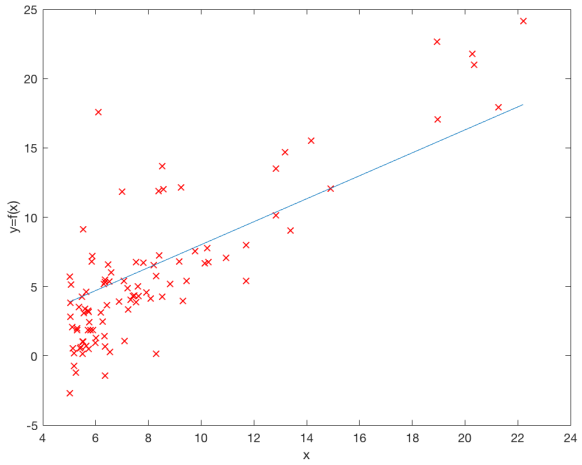
Task 1

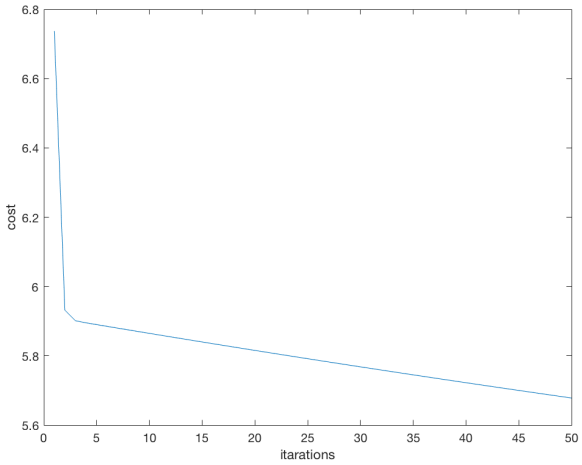
calculate_hypothesis.m

```
hypothesis = theta(1) * X(training_example, 1) + theta(2) * X(training_example, 2);
```

gradient_descent.m

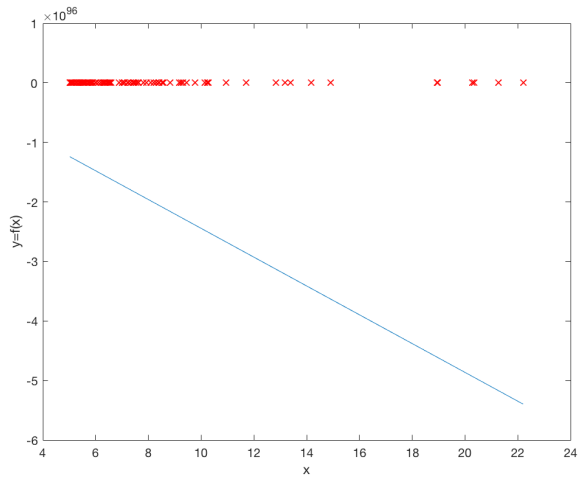
```
hypothesis = calculate_hypothesis(X, theta, i);
```

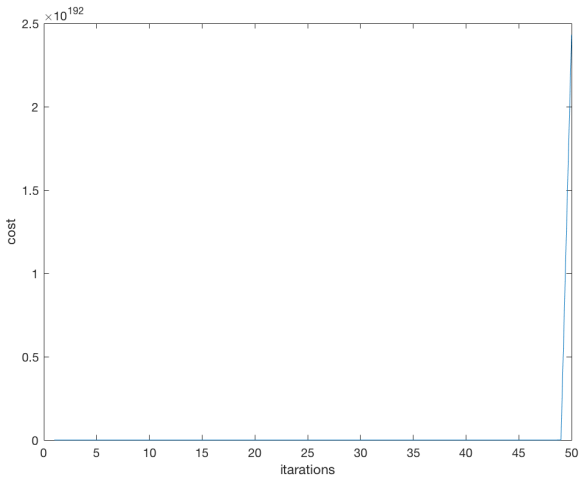




alpha = 1.0

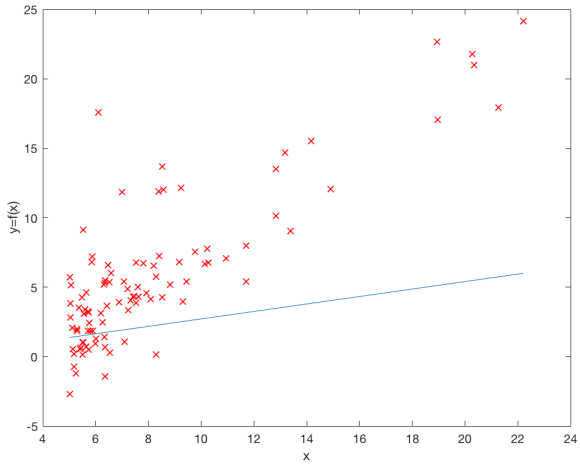
With a high learning rate, it is easy to overshoots and diverges the minimum of cost function so that it can't be converged.

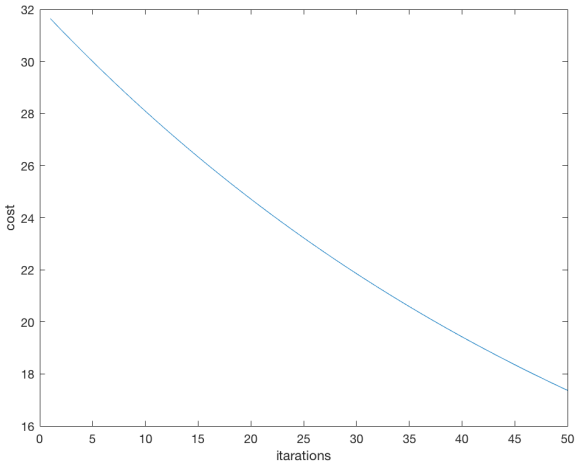




1. Linear Regression with One Variable

alpha = 0.00001
With a low learning rate, it is slow to find the minimum of cost function or even can't find it.





2. Linear Regression with Multiple Variables

Task 2

calculate_hypothesis.m

```
hypothesis = theta(1) * X(training_example, 1) + theta(2) * X(training_example, 2) + theta(3)
* X(training_example, 3);
```

gradient_descent.m

```
theta_2 = theta(3);

%update theta(3) and store in temporary variable theta_2
sigma = 0.0;

for i = 1:m
    hypothesis = calculate_hypothesis(X, theta, i);
    output = y(i);
    sigma = sigma + (hypothesis - output) * X(i, 3);
end

theta_2 = theta_2 - ((alpha * 1.0) / m) * sigma;

%update theta
theta = [theta_0, theta_1, theta_2];
```

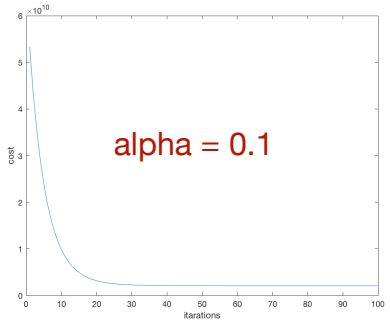
Print the theta values found at the end of the optimisation.

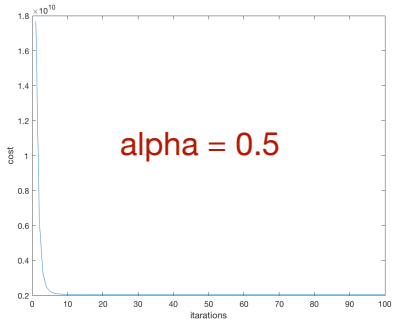
alpha	θ1	θ2	θ3
0.1	340403.617738031	109912.678488782	-5931.10911548232
0.5	340412.659574468	110631.050277122	-6649.47426909551
1.0	340412.659574468	110631.050278846	-6649.47427081980
1.2	340412.659574468	110631.049735430	-6649.47481423631
1.3	340412.659574468	99381.3194462383	-17899.2051034276
1.4	340412.659564112	-20444071819.0787	-20444189099.6033

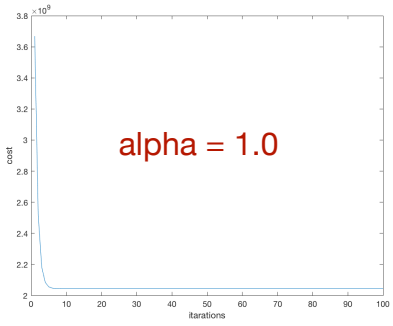
Changing learning rate (alpha) that can affect the convergence, and the optimisation is that modifying alpha to find the minimum of cost function. With a fixed iteration, I find the lowest value of cost function when alpha = 1.2 and the values of theta are 340412.659574468, 110631.049735430 and -6649.47481423631.

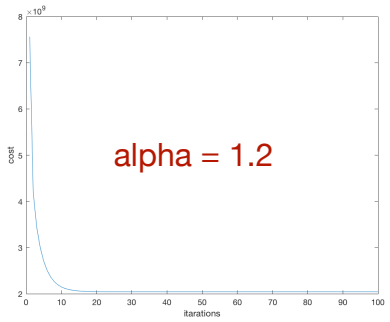
According to cost function figures below, once the learning rate is overshooting, the values of cost function will increase spectacularly. Therefore, the optimisation can be found before the boundary which has a increase of cost function .

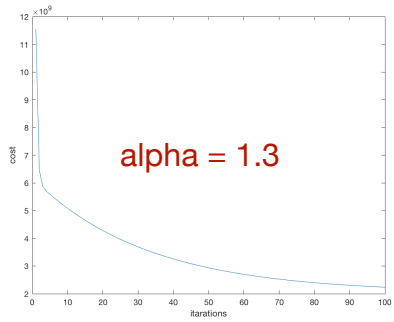
2. Linear Regression with Multiple Variables

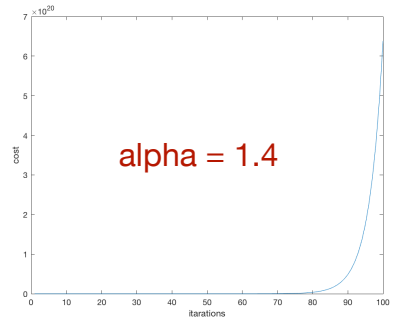












Use the trained theta values to make a prediction.

	the area of the house in square feet	number of bedrooms	predicted price
1	1650	3	293228.740256902
2	3000	4	472148.795259098

mllab2.m

```
%% use trained theta values to make predictions
Z = [1650,3; 3000,4];
rows = size(Z, 1);
Z = (Z - ones(rows, 1) * mean_vec) ./ (ones(rows, 1) * std_vec);
Z = [ones(rows, 1), Z];
predicted_price = zeros(rows, 1);
for i=1:size(Z, 1)
    predicted_price(i) = calculate_hypothesis(Z, t, i);
end
```

3. Regularised Linear Regression

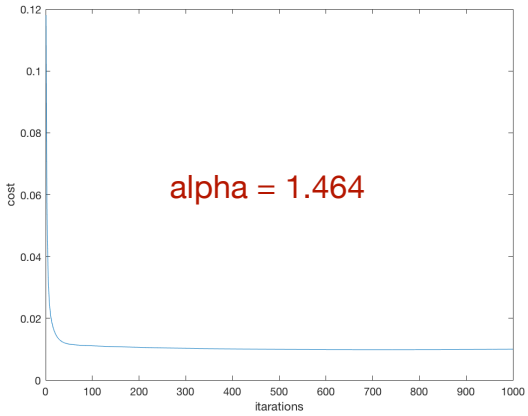
Task 3

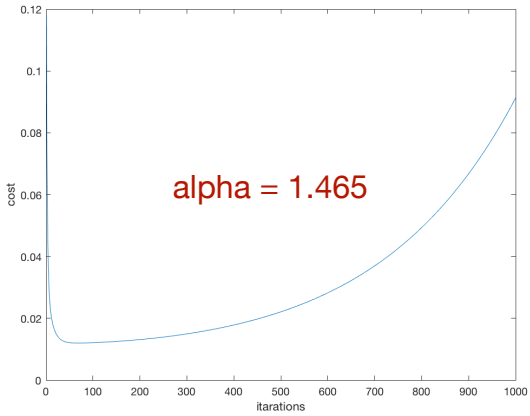
gradient_descent.m
```
function theta = gradient_descent(X, y, theta, alpha, iterations, l, do_plot)

cost_vector = [cost_vector; compute_cost_regularised(X, y, theta, l)];
```

mllab3.m
```
% initialise theta
theta = [1.0, 1.0, 1.0, 1.0, 1.0, 1.0];
```

gradient_descent.m
```
theta_0 = theta(1);
theta_1 = theta(2);
theta_2 = theta(3);
theta_3 = theta(4);
theta_4 = theta(5);
theta_5 = theta(6);

%update theta(6) and store in temporary variable theta_5
sigma = 0.0;

for i = 1:m
    hypothesis = calculate_hypothesis(X, theta, i);
    output = y(i);
    sigma = sigma + (hypothesis - output) * X(i, 6);
end

theta_5 = theta_5 * (1 - alpha * l / m) - ((alpha * 1.0) / m) * sigma;

%update theta
theta = [theta_0, theta_1, theta_2, theta_3, theta_4, theta_5];
```

Find the best value of alpha to use in order to optimise best.
alpha = 1.464



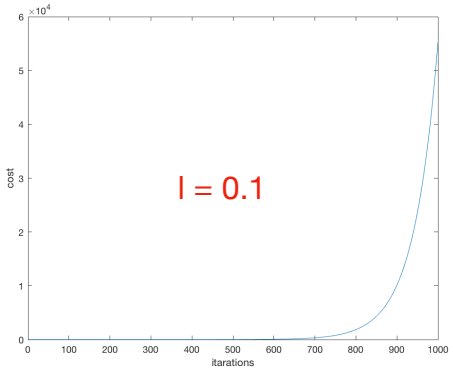


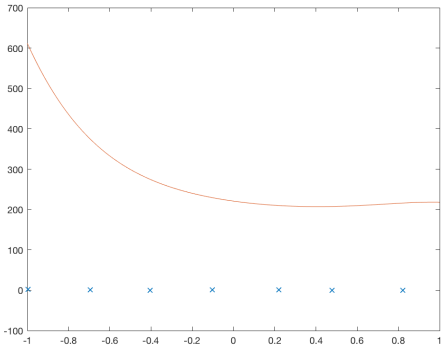
3. Regularised Linear Regression

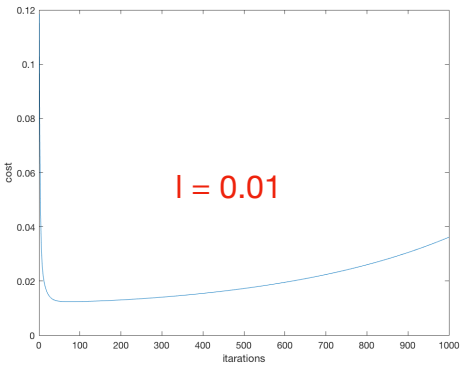
Experiment with different values of λ and see how this affects the shape of the hypothesis.

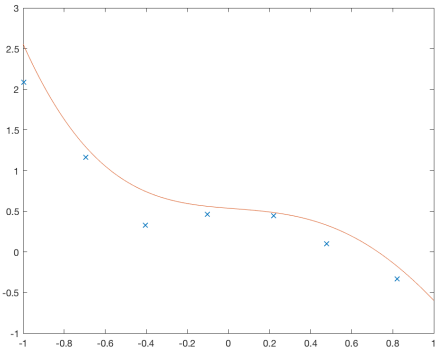
$$\theta_j = \theta_j(1 - \alpha\frac{\lambda}{m}) - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$$

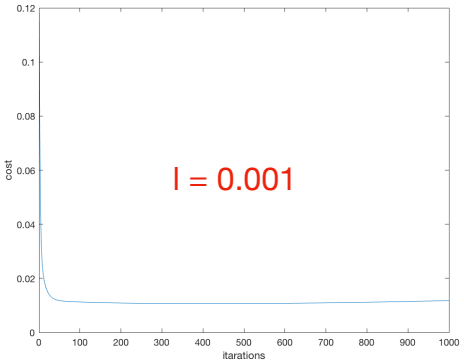
According to the regularisation formula, every value of theta will be effected by multiplying learning rate (alpha) by lambda. Once I set a high lambda, it will cause the value of cost function increase amazingly. Therefore, what I do is to decrease the value of lambda and find a better value of cost function.

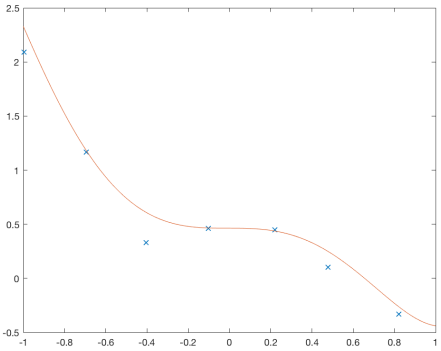












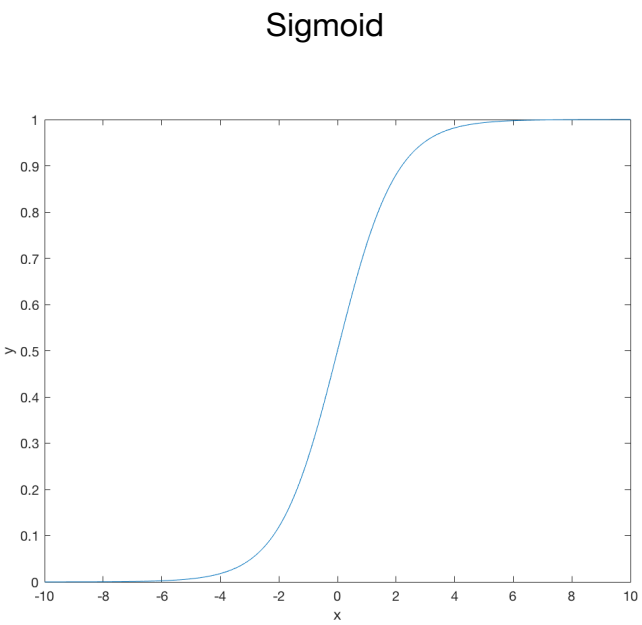
Logistic Regression and Neural Networks

1. Logistic Regression

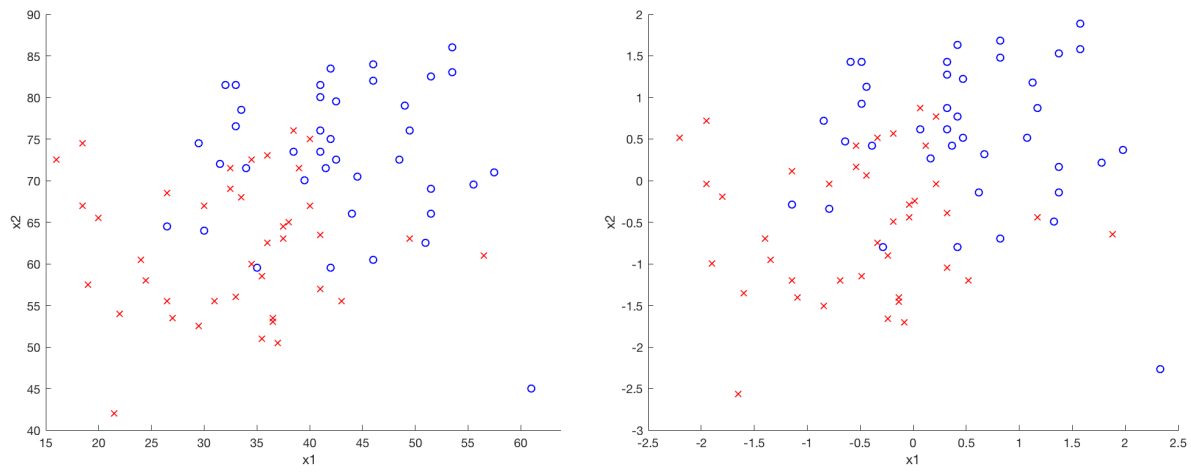
Task 1

sigmoid.m
```
output = 1 ./ (1 + exp(-z));
```

plot_sigmoid.m
```
h=figure(1); title('sigmoid');
```



Task 2



1.1. Cost function and gradient for logistic regression

Task 3

calculate_hypothesis.m

```
if size(X(training_example, :)) == size(theta)
    hypothesis = X(training_example, :) * theta';
end
```

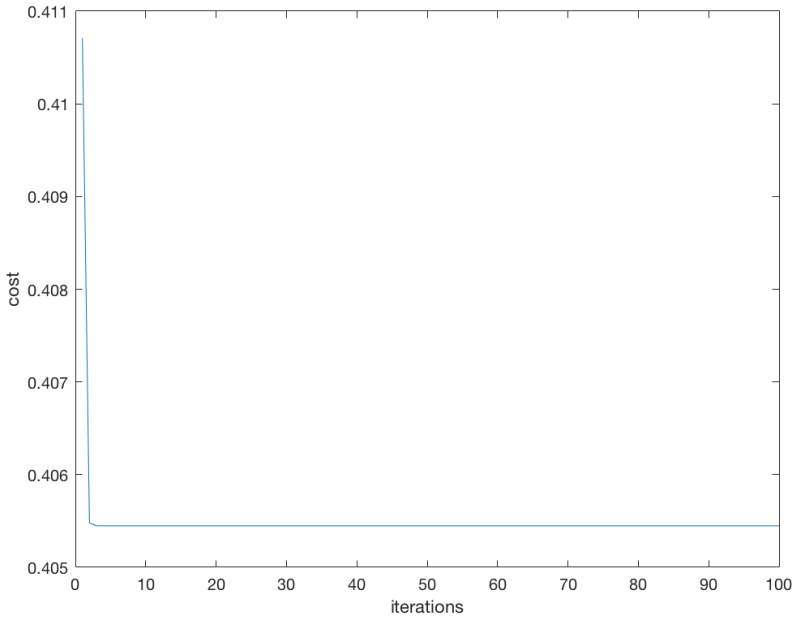
Task 4

compute_cost.m

```
cost = -output*log(hypothesis)-(1-output)*log(1-hypothesis);
```

Final error: 0.40545

Cost graph



Task 5

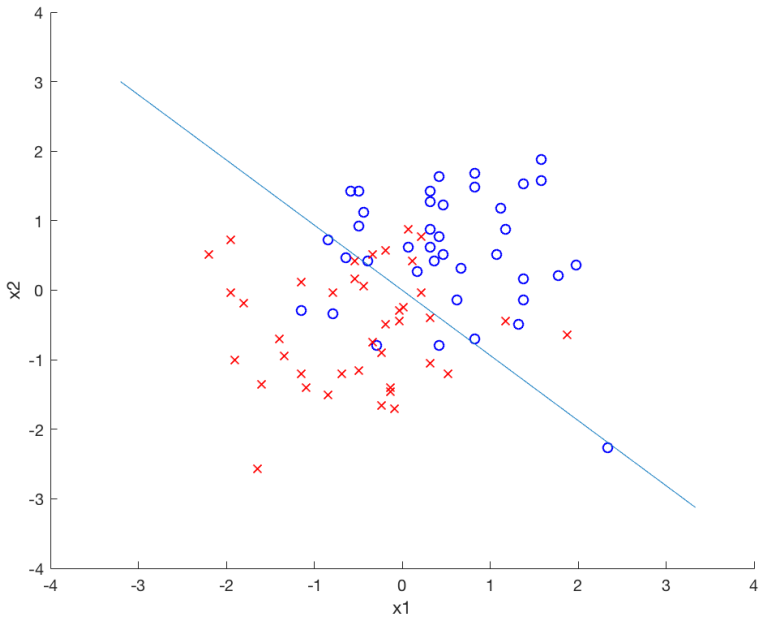
lab2_lr_ex1.m

```
plot_data_function(X,y)
plot_boundary(X,t)
```

plot_boundary.m

```
y1 = (-theta(2)*min_x1)/(theta(3));
y2 = (-theta(2)*max_x1)/(theta(3));
```

Data decision boundary graph



1.3. Non-linear features and overfitting

Task 6

	Training Error	Test Error
1	0.29649	0.63761
2	0.40755	0.67134
3	0.36283	0.54968
4	0.4675	0.40819
5	0.30031	0.82977

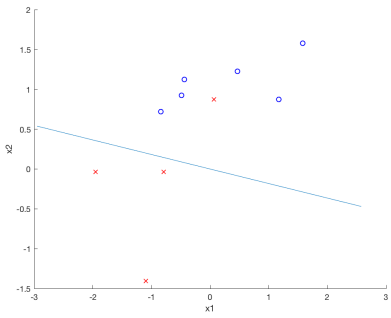
What is the general difference between the training and test error?

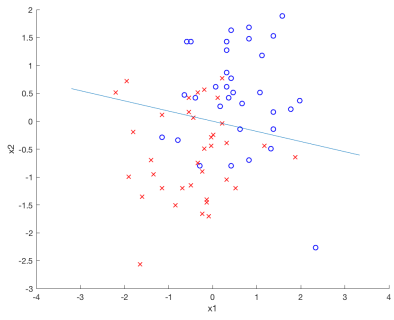
Mostly, the training error value is greater than the test error value. However, seldom it could happen the test error value is bigger. (row 4)

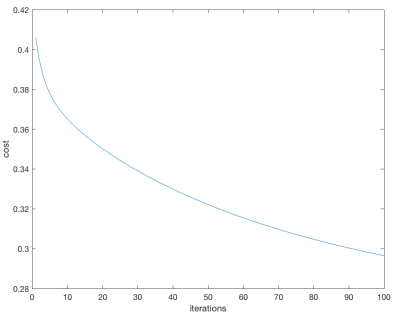
When does the training set generalise well?

According to the training model created on the training data, training set usually generalise well. However, if polynomials fits training data perfectly, it may cause overfitting and get a high test error. In the fifth row of table above, it may be a situation of overfitting as mentioned.

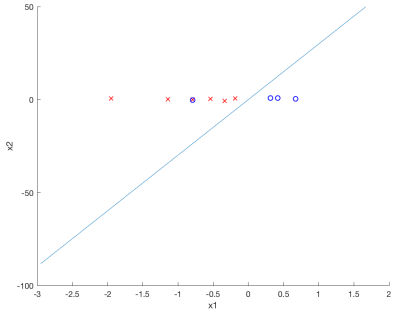
Good Generalisation

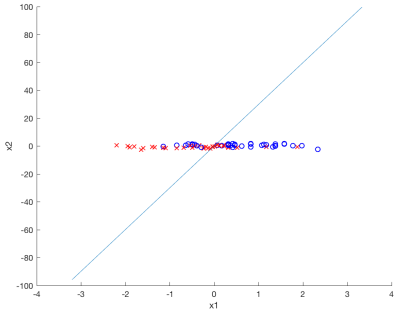


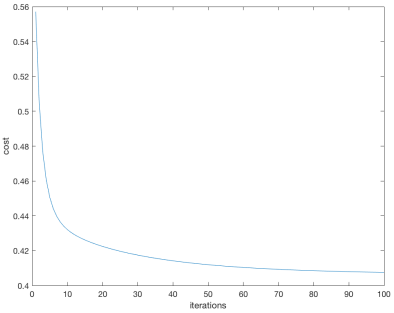




Bad Generalisation







1.3. Non-linear features and overfitting

lab2_lr_ex3a.m

```
theta=ones(1,6);

% here append x_2 * x_3 (remember that x_1 is the bias)
X = [X, (X(:,2) .* X(:,3))];
% here append x_2 * x_2 (remember that x_1 is the bias)
X = [X, (X(:,2) .* X(:,2))];
% here append x_3 * x_3 (remember that x_1 is the bias)
X = [X, (X(:,3) .* X(:,3))];
```

Task 7

How does the error compare to using the original features ?

Error:0.39537

Because this polynomial has six theta that can fit appropriately more than the other polynomial only has three theta.

Task 8

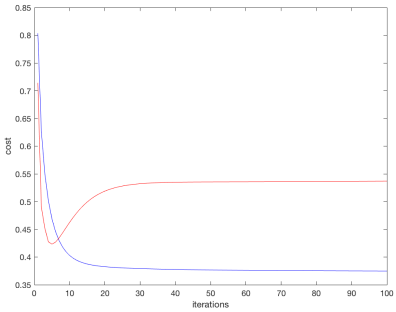
gradient_descent_training.m

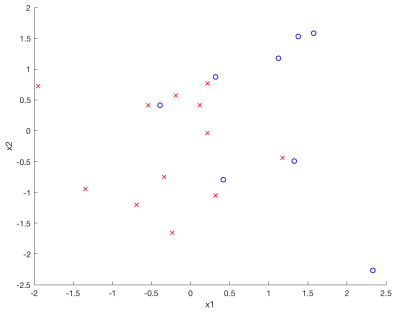
```
cost_array_training(it)=compute_cost(X, y, theta);
cost_array_test(it)=compute_cost(test_X, test_y, theta);
```

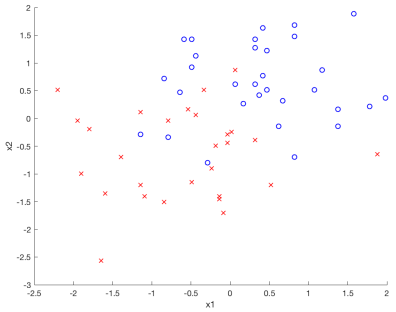
Number of training data	Training	Test
20	0.37472	0.53686
40	0.37789	0.4826
60	0.37138	0.54224

1.3. Non-linear features and overfitting

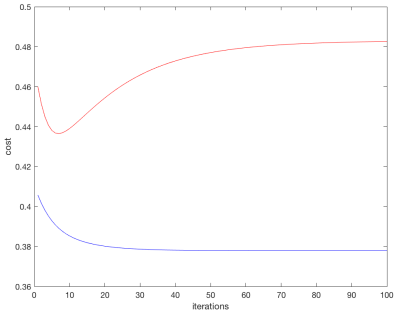
Number of training data: 20

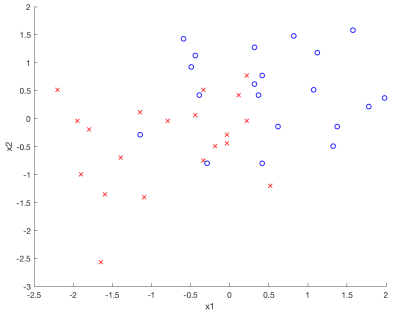


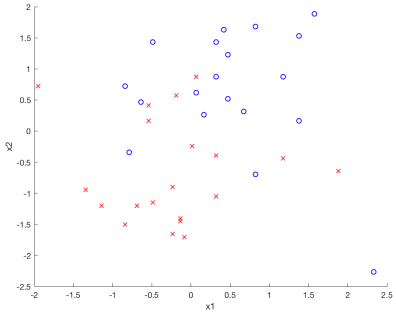




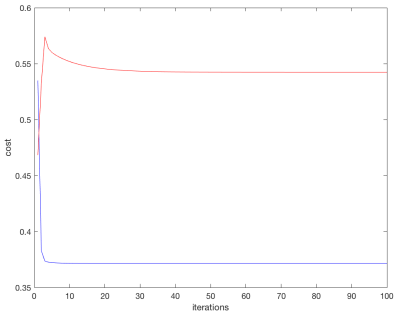
Number of training data: 40

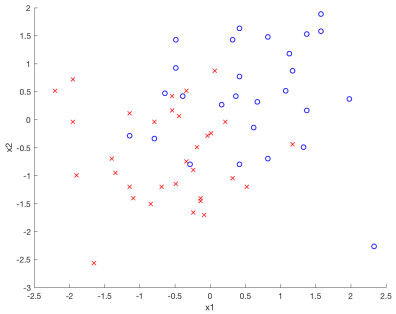


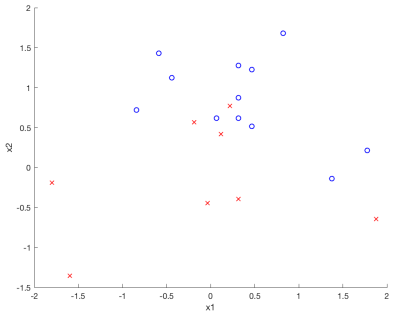




Number of training data: 60







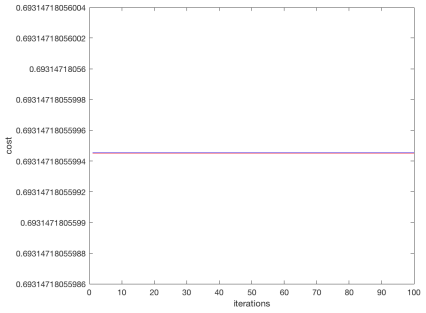
1.3. Non-linear features and overfitting

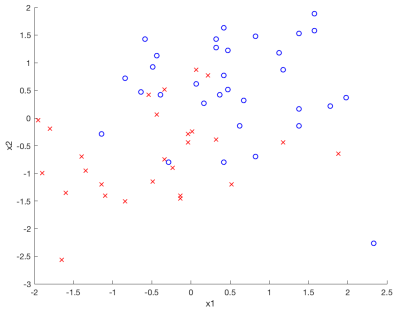
Add extra features (e.g. a third order polynomial) and analyse the effect.
lab2_lr_ex3b.m

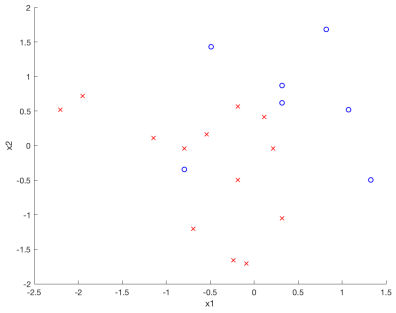
```
X = [X, (X(:,3) .* X(:,3) .* X(:,3))];
```

After adding a third order, the results of training and testing error are the same due to a higher order polynomial.

Third order polynomial
Training:0.69315 Test:0.69315







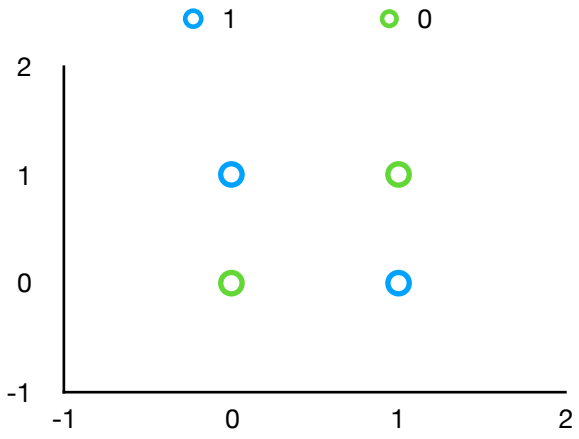
What happens when the cost function of the training set goes down but that of the test set goes up?

When it faces overfitting, it will cause the cost function of the training data decrease but the test data increase.

Task 9

With the aid of a diagram of the decision space, explain why a logistic regression unit cannot solve the XOR classification problem.

Logistic regression tries to find a linear separator. It can't be done when trying to draw a single line with ones on one side and zeros on the other.



2. Neural Network

sigmoid.m

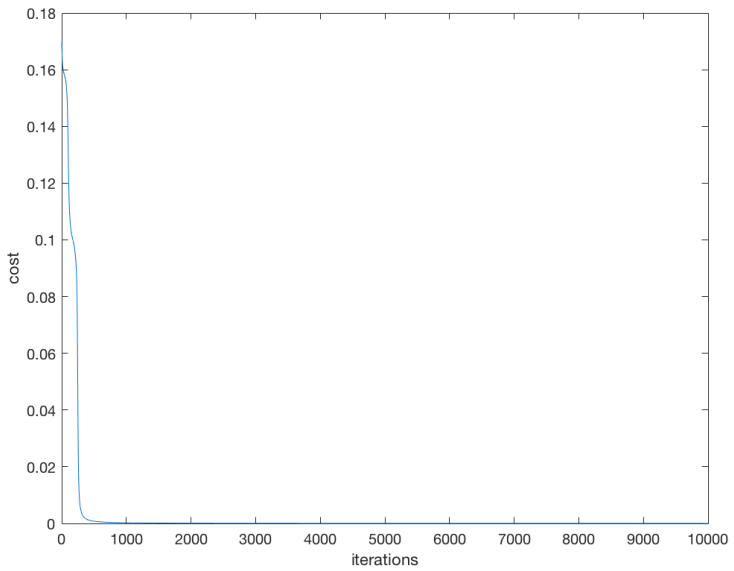
```
sigmoid_output = 1 ./ (1 + exp(-z));
```

Task 10

NeuralNetwork.m

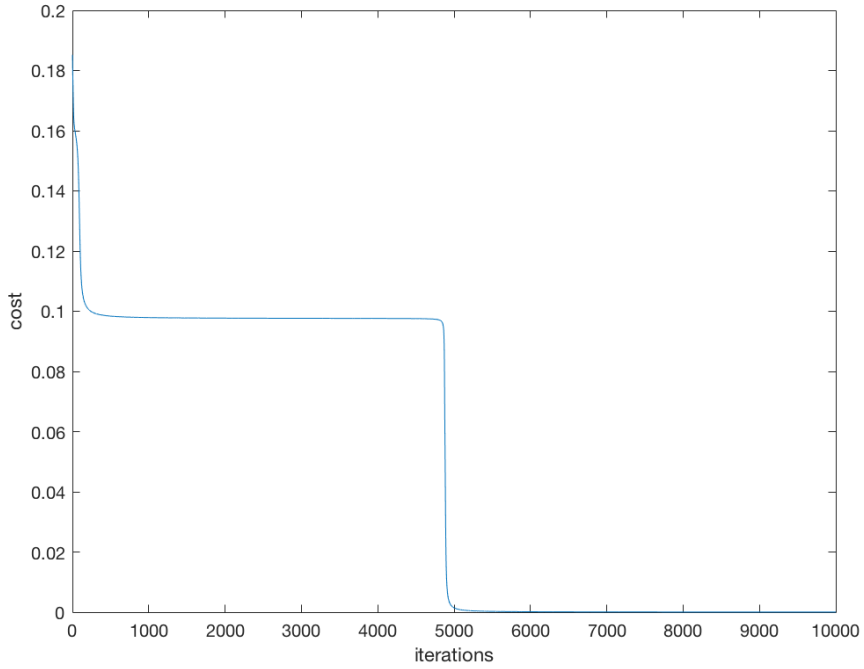
```
% hint... create a for loop here to iterate over the hidden neurons and for each
% hidden neuron create another for loop to iterate over the ouput neurons
hiddens=nn.hidden_neurons;
for i=1:length(hiddens)
    sigma = 0;
    for j=1:length(outputs)
        sigma = sigma + nn.output_weights(i,j) * output_deltas(j);
    end
    hidden_deltas(i) = sigmoid_derivative(hiddens(i)) * sigma;
end

% Step 4. update weights input --> hidden.
% hint, use a similar process to step 3, except iterate over the input neurons and hidden
deltas
nn.hidden_neurons = nn.hidden_neurons(2:end);
for i=1:length(inputs)
    for j=1:length(nn.hidden_neurons)
        nn.hidden_weights(i,j) = nn.hidden_weights(i,j) - (hidden_deltas(j) * inputs(i) *
learning_rate);
    end
end
```

Find the best learning rate.
alpha = 3.8



2. Neural Network

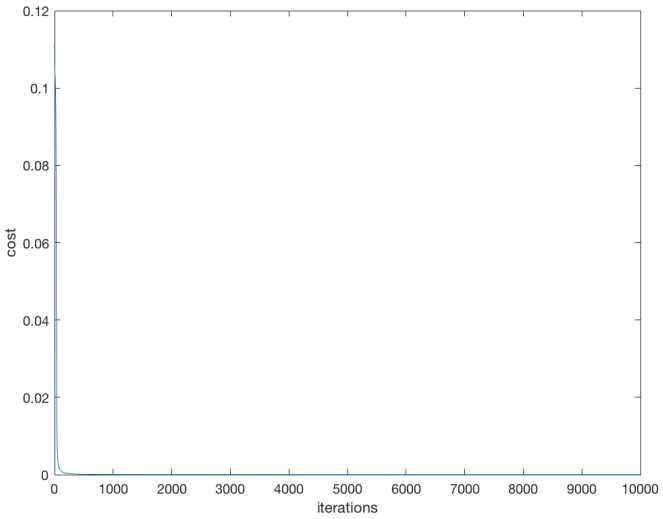
Get stuck in local optima.
alpha = 3.85
target output:0 actual output0.0067274
target output:1 actual output0.99271
target output:1 actual output0.99268
target output:0 actual output0.0093257



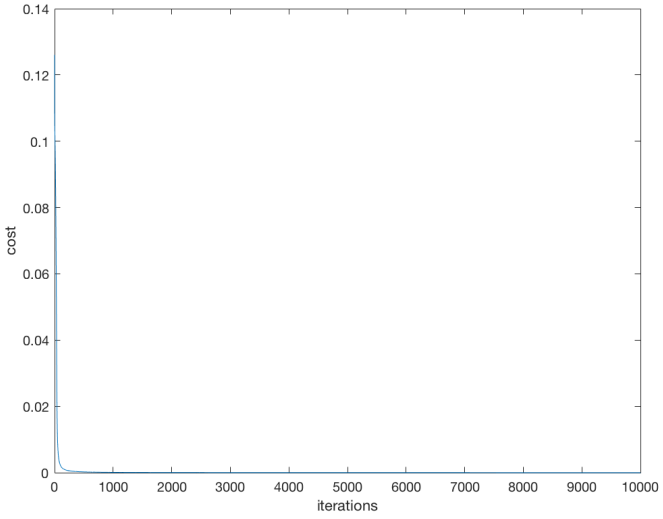
2.1. Implement backpropagation on XOR

Task 11

NOR
cost = 4.3547e-06
target output:1 actual output0.99589
target output:0 actual output0.0029403
target output:0 actual output0.0029554
target output:0 actual output0.00075941



AND
cost = 9.1851e-06
target output:0 actual output0.00073537
target output:0 actual output0.0036879
target output:0 actual output0.0039752
target output:1 actual output0.9934



2.2. Implement backpropagation on Iris

Task 12

How would accomplish this if we used a logistic regression unit?

It can be accomplished by multinomial logistic regression to generalises logistic regression to multiclass problems.

How is it different using a neural network?

Neural network can implement linear and logistic regression at the same time, and can do more complicate jobs by adding hidden layers.

Task 13

What are the differences for each number of hidden neurons?

Firstly, one of hidden neuron has the highest training and testing error. Next, the training and testing errors are similar when the number of hidden neurons are greater than one.

Which number do you think is the best to use?

In my opinion, according to table below, I think five of hidden neurons is the best to use because the training error is the lowest than others with similar testing error.

How well do you think that we have generalised?

Once I try to add the number of hidden neurons to 100 and I found the training and testing error is smaller than the values from 1 to 10 of neurons. Therefore, it means we can generalise well when we add the number of hidden neurons. However, it may cost lots of time to get a better result or face overfitting when adding too much hidden neurons, so we need to make an assignment among generalising well, overfitting and time spending.

	Error training	Error testing
1	18.065	18.9502
2	0.2849	4.9865
3	0.24607	4.9562
5	0.20881	4.9454
7	0.21246	4.9934
10	0.22266	4.838
100	0.18494	4.5376