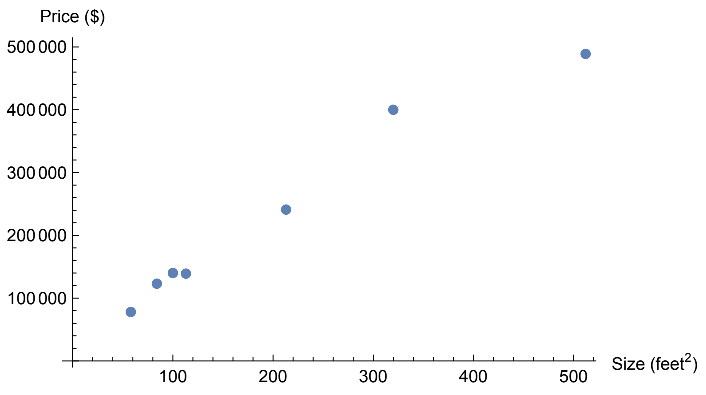
Week 1 – Stanford Machine Learning

Linear Regression

Let us create some random data, say housing sizes in square feet on **x** and price levels on **y**, and plot it.

```
trainX = {100, 320, 213, 512, 58, 84, 113};
trainY = {140 000, 400 000, 241 000, 489 000, 78 000, 123 000, 139 000};
reglp = ListPlot[Transpose@{trainX, trainY},
   AxesLabel → {"Size (feet^2)", "Price ($)"}];
Show[
 reglp]
```



Now, we define a linear hypothesis function since our data seems to allow for a linear approximation

```
Hyp[v_, x_] := v[[1]] + v[[2]] x;
```

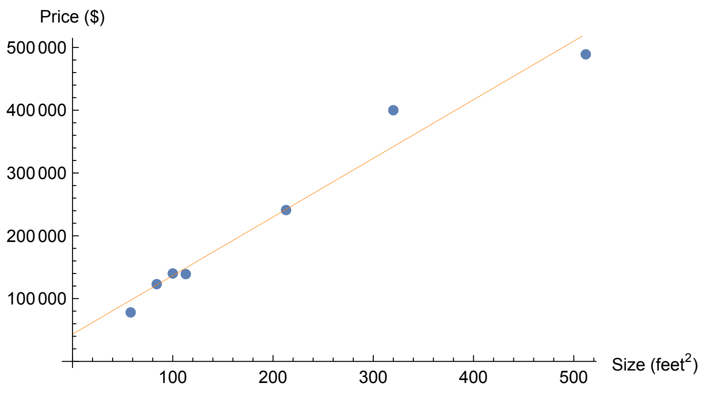
followed by the squared error cost function

```
Cost[v1_, v2_] :=        1
                  ─────────────────
                  2 Length[trainX]
   Total[(Hyp[{v1, v2}, #1] - #2)^2 & @@@ Transpose@{trainX, trainY}];
```

We then minimise the cost function, after which we plot the hypothesis function with the minimised values

```
regression = Minimize[Cost[v1, v2], {v1, v2}] // N
regpp := Plot[Hyp[{v1, v2} /. regression[[2]], x],
   {x, 0, Max[trainX]}, PlotStyle → {Orange, Thin}];
Show[reglp, regpp]
```

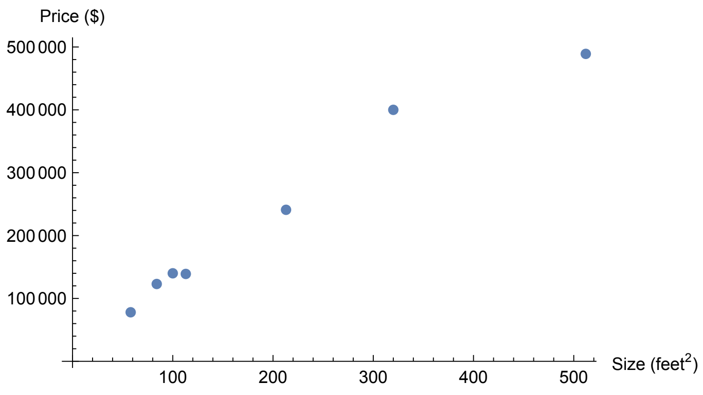
```
{3.49273 × 10^8, {v1 → 43 289.8, v2 → 933.551}}
```



Gradient Descent

Let us repeat the regression analysis by using gradient descent instead. First, we take the same data, and plot it

```
trainX = {100, 320, 213, 512, 58, 84, 113};
trainY = {140 000, 400 000, 241 000, 489 000, 78 000, 123 000, 139 000};
gdlp = ListPlot[Transpose@{trainX, trainY},
   AxesLabel → {"Size (feet^2)", "Price ($)"}];
Show[
 gdlp]
```



We define the same hypothesis and cost functions

```
Hyp[v_, x_] := v[[1]] + v[[2]] x;
Cost[v1_, v2_] := 1/(2 Length[trainX])
   Total[(Hyp[{v1, v2}, #1] - #2)^2 & @@@ Transpose@{trainX, trainY}];
```

And then we create the step function that returns the next step of the gradient descent algorithm. We also assign the learning rates. Note that because of the huge values on the pricing axis, the y values are particularly sensitive to the learning rate parameter. Any learning rate above 1×10^{-5}

will thus make the gradient descent algorithm overshoot. Instead of setting a very low learning rate parameter, as below, a better fix would be to represent the housing prices in terms of thousand dollars.

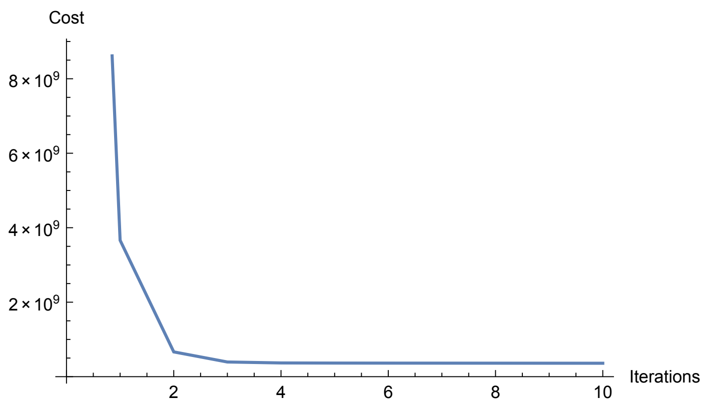
```
αv1 = 0.1;
αv2  = 0.00001;
Step[v1_, v2_] :=
  {v1 - αv1 D[Cost[x, y], x] /. {x → v1, y → v2},
   v2 - αv2 D[Cost[x, y], y] /. {x → v1, y → v2}};
```

To check that the step function works, we can plot the step function as a function of the iteration number to picture the learning rate. We do so by creating the learning function

```
Learning[x_] := Cost @@ Nest[Apply[Step, #] &, {1, 1}, x];
```

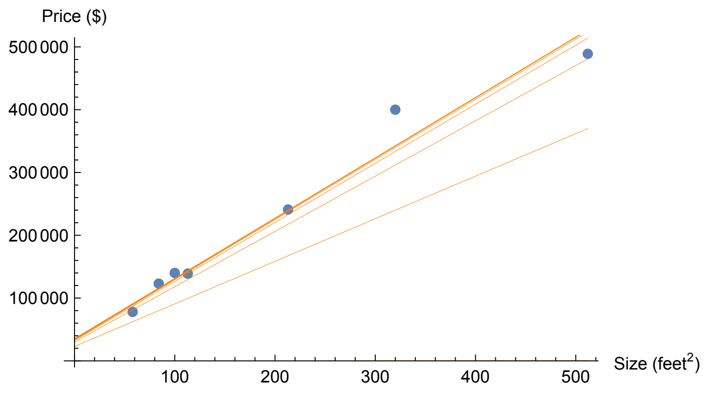
We do indeed learn, as seen in the plot below

```
learningRate = ListLinePlot[#, AxesLabel → {"Iterations", "Cost"}] &@
  Transpose@{Range[0, 10], Learning /@ Range[0, 10]}
```



We then step through the algorithm ten times and plot all the lines of all intermediate steps to see how the regression converges towards the optimum

```
gd = NestList[Apply[Step, #] &, {1, 1}, 10];
gdpp = Show[
   Plot[Hyp[#1, x], {x, 0, Max[trainX]}, PlotStyle → {Orange, Thin}] & /@ gd];
Show[
 gdlp,
 gdpp]
```


Price ($)
Size (feet²)