Error-based Learning I

Bojan Božić

TU Dublin

Summer 2021

Big Idea

- A **paramaterised** prediction model is initialised with a set of random parameters and an error function is used to judge how well this initial model performs when making predictions for instances in a training dataset.
- Based on the value of the error function the parameters are iteratively adjusted to create a more and more accurate model.

Fundamentals

Table: The **office rentals dataset**: a dataset that includes office rental prices and a number of descriptive features for 10 Dublin city-centre offices.

ID	SIZE	FLOOR	BROADBAND RATE	ENERGY RATING	RENTAL PRICE
1	500	4	8	C	320
2	550	7	50	A	380
3	620	9	7	A	400
4	630	5	24	B	390
5	665	8	100	C	385
6	700	4	8	B	410
7	770	10	7	B	480
8	880	12	50	A	600
9	920	14	8	C	570
10	1,000	9	24	B	620

Table: The **office rentals dataset**: a dataset that includes office rental prices and a number of descriptive features for 10 Dublin city-centre offices.

ID	SIZE	RENTAL PRICE
1	500	320
2	550	380
3	620	400
4	630	390
5	665	385
6	700	410
7	770	480
8	880	600
9	920	570
10	1,000	620

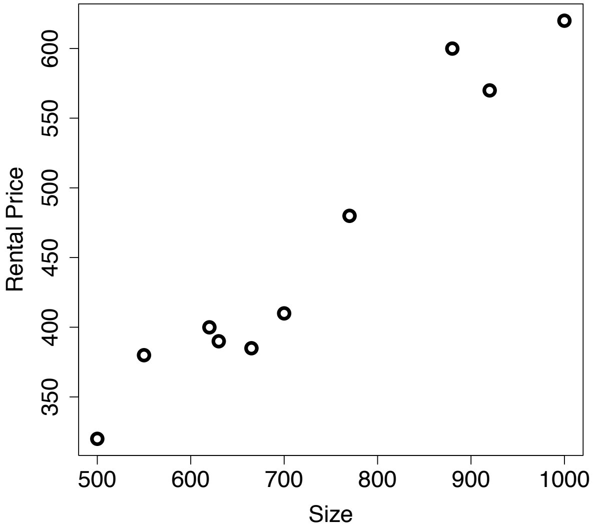


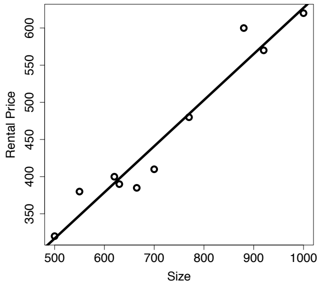
Figure: A scatter plot of the SIZE and RENTAL PRICE features from the office rentals dataset.

- From the scatter plot it appears that there is a linear relationship between the SIZE and RENTAL PRICE.
- The equation of a line can be written as:

$$y = mx + b \tag{1}$$

- The scatter plot below shows the same scatter plot as shown in Figure 1 [8] with a simple linear model added to capture the relationship between office sizes and office rental prices.
- This model is:

$$\text{RENTAL PRICE} = 6.47 + 0.62 \times \text{SIZE}$$



$$\text{RENTAL PRICE} = 6.47 + 0.62 \times \text{SIZE}$$

- Using this model determine the expected rental price of the 730 square foot office:

Simple Linear Regression

$$\text{RENTAL PRICE} = 6.47 + 0.62 \times \text{SIZE}$$

- Using this model determine the expected rental price of the 730 square foot office:

$$\begin{aligned}\text{RENTAL PRICE} &= 6.47 + 0.62 \times 730\\ &= 459.07\end{aligned}$$

$$\mathbb{M}_{\mathbf{w}}(d) = \mathbf{w}[0] + \mathbf{w}[1] \times \mathbf{d}[1] \tag{2}$$

Measuring Error

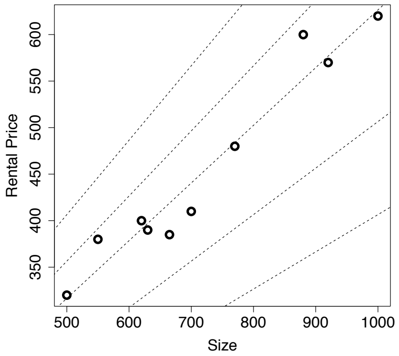


Figure: A scatter plot of the SIZE and RENTAL PRICE features from the office rentals dataset. A collection of possible simple linear regression models capturing the relationship between these two features are also shown. For all models **w**[0] is set to 6.47. From top to bottom the models use 0.4, 0.5, 0.62, 0.7 and 0.8 respectively for **w**[1].

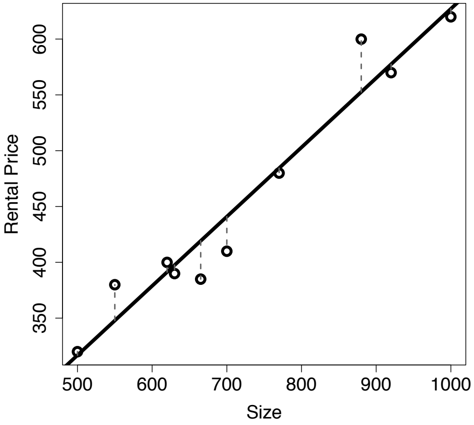


Figure: A scatter plot of the SIZE and RENTAL PRICE features from the office rentals dataset showing a candidate prediction model (with $\mathbf{w}[0] = 6.47$ and $\mathbf{w}[1] = 0.62$) and the resulting errors.

Measuring Error

$$L_2(\mathbb{M}_{\mathbf{w}}, \mathcal{D}) = \frac{1}{2}\sum_{i=1}^{n}(t_i - \mathbb{M}_{\mathbf{w}}(\mathbf{d}_i[1]))^2 \quad (3)$$

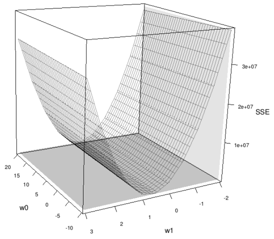
$$= \frac{1}{2}\sum_{i=1}^{n}(t_i - (\mathbf{w}[0] + \mathbf{w}[1] \times \mathbf{d}_i[1]))^2 \quad (4)$$

Table: Calculating the sum of squared errors for the candidate model (with $\mathbf{w}[0] = 6.47$ and $\mathbf{w}[1] = 0.62$) making predictions for the the office rentals dataset.

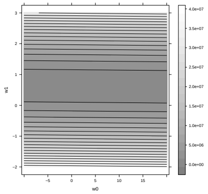
ID	RENTAL PRICE	Model Prediction	Error Error	Squared Error
1	320	316.79	3.21	10.32
2	380	347.82	32.18	1,035.62
3	400	391.26	8.74	76.32
4	390	397.47	-7.47	55.80
5	385	419.19	-34.19	1,169.13
6	410	440.91	-30.91	955.73
7	480	484.36	-4.36	19.01
8	600	552.63	47.37	2,243.90
9	570	577.46	-7.46	55.59
10	620	627.11	-7.11	50.51
			Sum	**5,671.64**
			Sum of squared errors (Sum/2)	**2,835.82**

Error Surfaces

- For every possible combination of weights, **w**[0] and **w**[1], there is a corresponding sum of squared errors value that can be joined together to make a surface.



(a)



(b)

Figure: (a) A 3D surface plot and (b) a contour plot of the error surface generated by plotting the sum of squared errors value for the office rentals training set for each possible combination of values for **w**[0] (from the range $[-10, 20]$) and **w**[1] (from the range $[-2, 3]$).

- The *x*-*y* plane is known as a **weight space** and the surface is known as an **error surface**.
- The model that best fits the training data is the model corresponding to the lowest point on the error surface.

- Using Equation (4)[16] we can formally define this point on the error surface as the point at which:

$$\frac{\partial}{\partial \mathbf{w}[0]} \frac{1}{2} \sum_{i=1}^{n} (t_i - (\mathbf{w}[0] + \mathbf{w}[1] \times \mathbf{d}_i[1]))^2 = 0 \tag{5}$$

and

$$\frac{\partial}{\partial \mathbf{w}[1]} \frac{1}{2} \sum_{i=1}^{n} (t_i - (\mathbf{w}[0] + \mathbf{w}[1] \times \mathbf{d}_i[1]))^2 = 0 \tag{6}$$

- There are a number of different ways to find this point.
- We will describe a **guided search** approach known as the **gradient descent** algorithm.

Standard Approach: Multivariate Linear Regression with Gradient Descent

Multivariate Linear Regression

Table: A dataset that includes office rental prices and a number of descriptive features for 10 Dublin city-center offices.

ID	SIZE	FLOOR	BROADBAND RATE	ENERGY RATING	RENTAL PRICE
1	500	4	8	C	320
2	550	7	50	A	380
3	620	9	7	A	400
4	630	5	24	B	390
5	665	8	100	C	385
6	700	4	8	B	410
7	770	10	7	B	480
8	880	12	50	A	600
9	920	14	8	C	570
10	1,000	9	24	B	620

- We can define a multivariate linear regression model as:

$$\mathbb{M}_{\mathbf{w}}(\mathbf{d}) = \mathbf{w}[0] + \mathbf{w}[1] \times \mathbf{d}[1] + \cdots + \mathbf{w}[m] \times \mathbf{d}[m] \quad (7)$$

$$= \mathbf{w}[0] + \sum_{j=1}^{m} \mathbf{w}[j] \times \mathbf{d}[j] \quad (8)$$

- We can make Equation (8)[23] look a little neater by inventing a dummy descriptive feature, $\mathbf{d}[0]$, that is always equal to 1:

$$\begin{aligned}
\mathbb{M}_{\mathbf{w}}(\mathbf{d}) &= \mathbf{w}[0] \times \mathbf{d}[0] + \mathbf{w}[1] \times \mathbf{d}[1] + \ldots + \mathbf{w}[m] \times \mathbf{d}[m] && (9)\\
&= \sum_{j=0}^{m} \mathbf{w}[j] \times \mathbf{d}[j] && (10)\\
&= \mathbf{w} \cdot \mathbf{d} && (11)
\end{aligned}$$

- The sum of squared errors loss function, L_2, definition that we gave in Equation (4)[16] changes only very slightly to reflect the new regression equation:

$$L_2(\mathbb{M}_{\mathbf{w}}, \mathcal{D}) = \frac{1}{2}\sum_{i=1}^{n}(t_i - \mathbb{M}_{\mathbf{w}}(\mathbf{d}_i))^2 \quad (12)$$

$$= \frac{1}{2}\sum_{i=1}^{n}(t_i - (\mathbf{w} \cdot \mathbf{d}_i))^2 \quad (13)$$

- This multivariate model allows us to include all but one of the descriptive features in Table 3 [17] in a regression model to predict office rental prices.
- The resulting multivariate regression model equation is:

$$\begin{aligned}\textsc{Rental Price} = \mathbf{w}[0] &+ \mathbf{w}[1] \times \textsc{Size} + \mathbf{w}[2] \times \textsc{Floor} \\ &+ \mathbf{w}[3] \times \textsc{Broadband Rate}\end{aligned}$$

- We will see in the next section how the best-fit set of weights for this equation are found, but for now we will set:
  - $\mathbf{w}[0] = -0.1513$,
  - $\mathbf{w}[1] = 0.6270$,
  - $\mathbf{w}[2] = -0.1781$,
  - $\mathbf{w}[3] = 0.0714$.
- This means that the model is rewritten as:

$$\begin{aligned}\text{RENTAL PRICE} = -0.1513 \quad &+ \quad 0.6270 \times \text{SIZE}\\ &- \quad 0.1781 \times \text{FLOOR}\\ &+ \quad 0.0714 \times \text{BROADBAND RATE}\end{aligned}$$

- Using this model:

$$\begin{aligned}\textsc{Rental Price} = -0.1513 & + 0.6270 \times \textsc{Size} \\ & - 0.1781 \times \textsc{Floor} \\ & + 0.0714 \times \textsc{Broadband Rate}\end{aligned}$$

- we can, for example, predict the expected rental price of a 690 square foot office on the 11*th* floor of a building with a broadband rate of 50 Mb per second as:

$$\textsc{Rental Price} = ?$$

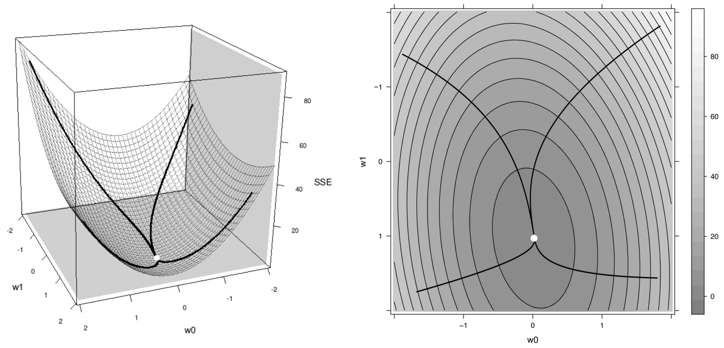
- Using this model:

$$\begin{aligned}\textsc{Rental Price} = -0.1513 &+ 0.6270 \times \textsc{Size}\\ &- 0.1781 \times \textsc{Floor}\\ &+ 0.0714 \times \textsc{Broadband Rate}\end{aligned}$$

- we can, for example, predict the expected rental price of a 690 square foot office on the 11^{th} floor of a building with a broadband rate of 50 Mb per second as:

$$\begin{aligned}\textsc{Rental Price} &= -0.1513 + 0.6270 \times 690\\ &\quad -0.1781 \times 11 + 0.0714 \times 50\\ &= 434.0896\end{aligned}$$

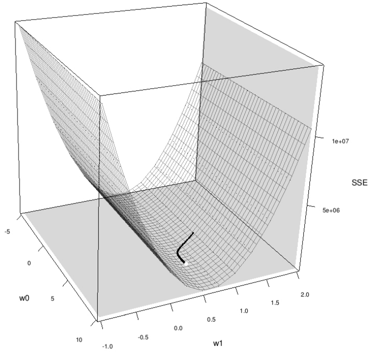
Gradient Descent



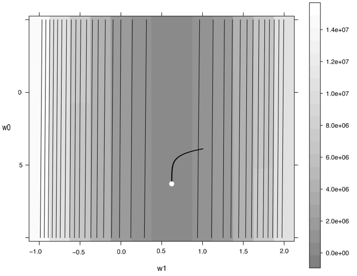
(a) (b)

Figure: (a) A 3D surface plot and (b) a contour plot of the same error surface. The lines indicate the path that the gradient decent algorithm would take across this error surface from different starting positions to the global minimum - marked as the white dot in the centre.

- The journey across the error surface that is taken by the gradient descent algorithm when training the simple version of the office rentals example - involving just SIZE and RENTAL PRICE.



(a)



(b)

Figure: (a) A 3D surface plot and (b) a contour plot of the error surface for the office rentals dataset showing the path that the gradient descent algorithm takes towards the best fit model.

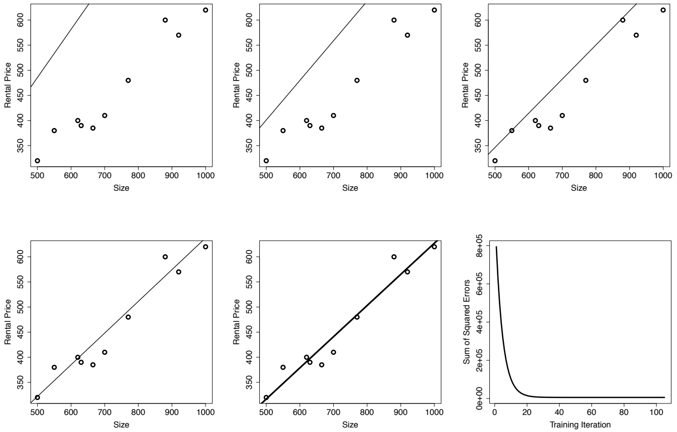


Figure: A selection of the simple linear regression models developed during the gradient descent process for the office rentals dataset. The final panel shows the sum of squared error values generated during the gradient descent process.

```
Require: set of training instances $\mathcal{D}$
Require: a learning rate $\alpha$ that controls how quickly the
    algorithm converges
Require: a function, errorDelta, that determines the direction
    in which to adjust a given weight, $\mathbf{w}[j]$, so as to move down
    the slope of an error surface determined by the dataset, $\mathcal{D}$
Require: a convergence criterion that indicates that the
    algorithm has completed
1: $\mathbf{w} \leftarrow$ random starting point in the weight space
2: repeat
3:     for each $\mathbf{w}[j]$ in $\mathbf{w}$ do
4:         $\mathbf{w}[j] \leftarrow \mathbf{w}[j] + \alpha \times \textbf{errorDelta}(\mathcal{D}, \mathbf{w}[j])$
5:     end for
6: until convergence occurs
```

- The gradient descent algorithm for training multivariate linear regression models.

Gradient Descent

- The most important part to the gradient descent algorithm is Line Rule 4 on which the weights are updated.

$$\mathbf{w}[j] \leftarrow \mathbf{w}[j] + \alpha \times \mathbf{errorDelta}(\mathcal{D}, \mathbf{w}[j])$$

- Each weight is considered independently and for each one a small adjustment is made by adding a small **delta** value to the current weight, **w**[*j*].
- This adjustment should ensure that the change in the weight leads to a move *downwards* on the error surface.

- Imagine for a moment that our training dataset, $\mathcal{D}$ contains just one training example: $(\mathbf{d}, t)$
- The gradient of the error surface is given as the partial derivative of L_2 with respect to each weight, $\mathbf{w}[j]$:

$$\begin{aligned}
\frac{\partial}{\partial \mathbf{w}[j]} L_2(\mathbb{M}_{\mathbf{w}}, \mathcal{D}) &= \frac{\partial}{\partial \mathbf{w}[j]} \left(\frac{1}{2} (t - \mathbb{M}_{\mathbf{w}}(\mathbf{d}))^2 \right) && (14) \\
&= (t - \mathbb{M}_{\mathbf{w}}(\mathbf{d})) \times \frac{\partial}{\partial \mathbf{w}[j]} (t - \mathbb{M}_{\mathbf{w}}(\mathbf{d})) && (15) \\
&= (t - \mathbb{M}_{\mathbf{w}}(\mathbf{d})) \times \frac{\partial}{\partial \mathbf{w}[j]} (t - (\mathbf{w} \cdot \mathbf{d})) && (16) \\
&= (t - \mathbb{M}_{\mathbf{w}}(\mathbf{d})) \times -\mathbf{d}[j] && (17)
\end{aligned}$$

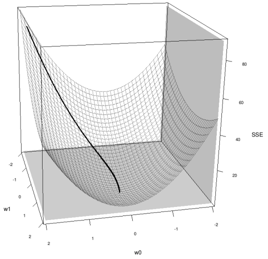
- Adjusting the calculation to take into account multiple training instances:

$$\frac{\partial}{\partial \mathbf{w}[j]} L_2(\mathbb{M}_{\mathbf{w}}, \mathcal{D}) = \sum_{i=1}^{n} \left((t_i - \mathbb{M}_{\mathbf{w}}(\mathbf{d}_i)) \times \mathbf{d}_i[j] \right)$$

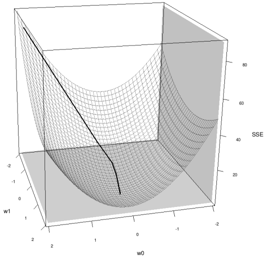
- We use this equation to define the **errorDelta** in our gradient descent algorithm.

$$\mathbf{w}[j] \leftarrow \mathbf{w}[j] + \alpha \underbrace{\sum_{i=1}^{n} \left((t_i - \mathbb{M}_{\mathbf{w}}(\mathbf{d}_i)) \times d_i[j] \right)}_{errorDelta(\mathcal{D}, \mathbf{w}[j])}$$

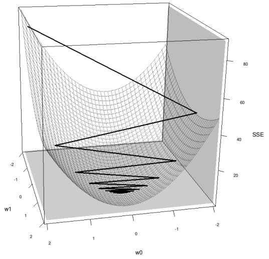
- The learning rate, α, determines the size of the adjustment made to each weight at each step in the process.
- Unfortunately, choosing learning rates is not a well defined science.
- Most practitioners use rules of thumb and trial and error.



(a)



(b)



(c)

Figure: Plots of the journeys made across the error surface for the simple office rentals prediction problem for different learning rates: (a) a very small learning rate (0.002), (b) a medium learning rate (0.08) and (c) a very large learning rate (0.18).

- A typical range for learning rates is $[0.00001, 10]$
- Based on empirical evidence, choosing random initial weights uniformly from the range $[-0.2, 0.2]$ tends to work well.

- We are now in a position to build a linear regression model that uses all of the continuous descriptive features in the office rentals dataset.
- The general structure of the model is:

$$\begin{aligned}\textsc{Rental Price} = \mathbf{w}[0] &+ \mathbf{w}[1] \times \textsc{Size} + \mathbf{w}[2] \times \textsc{Floor} \\ &+ \mathbf{w}[3] \times \textsc{Broadband Rate}\end{aligned}$$

Table: The **office rentals dataset**: a dataset that includes office rental prices and a number of descriptive features for 10 Dublin city-centre offices.

ID	SIZE	FLOOR	BROADBAND RATE	ENERGY RATING	RENTAL PRICE
1	500	4	8	C	320
2	550	7	50	A	380
3	620	9	7	A	400
4	630	5	24	B	390
5	665	8	100	C	385
6	700	4	8	B	410
7	770	10	7	B	480
8	880	12	50	A	600
9	920	14	8	C	570
10	1,000	9	24	B	620

A Worked Example

- For this example let's assume that:
  - $\alpha = 0.00000002$

Initial Weights

w[0]:	-0.146	**w**[1]:	0.185	**w**[2]:	-0.044	**w**[3]:	0.119

A Worked Example

Iteration 1

ID	Rental Price	Pred.	Error	Squared Error	errorDelta($\mathcal{D}$, **w**[**i**]) **w**[0]	**w**[1]	**w**[2]	**w**[3]
1	320	93.26	226.74	51411.08	226.74	113370.05	906.96	1813.92
2	380	107.41	272.59	74307.70	272.59	149926.92	1908.16	13629.72
3	400	115.15	284.85	81138.96	284.85	176606.39	2563.64	1993.94
4	390	119.21	270.79	73327.67	270.79	170598.22	1353.95	6498.98
5	385	134.64	250.36	62682.22	250.36	166492.17	2002.91	25036.42
6	410	130.31	279.69	78226.32	279.69	195782.78	1118.76	2237.52
7	480	142.89	337.11	113639.88	337.11	259570.96	3371.05	2359.74
8	600	168.32	431.68	186348.45	431.68	379879.24	5180.17	21584.05
9	570	170.63	399.37	159499.37	399.37	367423.83	5591.23	3194.99
10	620	187.58	432.42	186989.95	432.42	432423.35	3891.81	10378.16
			Sum	1067571.59	3185.61	2412073.90	27888.65	88727.43
Sum of squared errors (Sum/2)				533785.80				

A Worked Example

$$\mathbf{w}[j] \leftarrow \mathbf{w}[j] + \alpha \underbrace{\sum_{i=1}^{n} ((t_i - \mathbb{M}_{\mathbf{w}}(\mathbf{d}_i)) \times d_i[j])}_{errorDelta(\mathcal{D},\mathbf{w}[j])}$$

Initial Weights

w[0]:	-0.146	**w**[1]:	0.185	**w**[2]:	-0.044	**w**[3]:	0.119

Example

$$\mathbf{w}[1] \leftarrow 0.185 + 0.00000002 \times 2,412,074 = 0.23324148$$

New Weights (Iteration 1)

w[0]:	-0.146	**w**[1]:	0.233	**w**[2]:	-0.043	**w**[3]:	0.121

A Worked Example

Iteration 2

ID	RENTAL PRICE	Pred.	Error	Squared Error	errorDelta($\mathcal{D}$, **w**[**i**]) **w**[0]	**w**[1]	**w**[2]	**w**[3]
1	320	117.40	202.60	41047.92	202.60	101301.44	810.41	1620.82
2	380	134.03	245.97	60500.69	245.97	135282.89	1721.78	12298.44
3	400	145.08	254.92	64985.12	254.92	158051.51	2294.30	1784.45
4	390	149.65	240.35	57769.68	240.35	151422.55	1201.77	5768.48
5	385	166.90	218.10	47568.31	218.10	145037.57	1744.81	21810.16
6	410	164.10	245.90	60468.86	245.90	172132.91	983.62	1967.23
7	480	180.06	299.94	89964.69	299.94	230954.68	2999.41	2099.59
8	600	210.87	389.13	151424.47	389.13	342437.01	4669.60	19456.65
9	570	215.03	354.97	126003.34	354.97	326571.94	4969.57	2839.76
10	620	187.58	432.42	186989.95	432.42	432423.35	3891.81	10378.16
			Sum	886723.04	2884.32	2195615.84	25287.08	80023.74
Sum of squared errors (Sum/2)				443361.52				

$$\mathbf{w}[j] \leftarrow \mathbf{w}[j] + \alpha \underbrace{\sum_{i=1}^{n} \left(\left(t_i - \mathbb{M}_{\mathbf{w}}(\mathbf{d}_i) \right) \times d_i[j] \right)}_{errorDelta(\mathcal{D}, \mathbf{w}[j])}$$

Initial Weights (Iteration 2)

w[0]:	-0.146	**w**[1]:	0.233	**w**[2]:	-0.043	**w**[3]:	0.121

Exercise

$$\mathbf{w}[1] \leftarrow ?, \alpha = 0.00000002$$

New Weights (Iteration 2)

w[0]:	?	**w**[1]:	?	**w**[2]:	?	**w**[3]:	?

$$\mathbf{w}[j] \leftarrow \mathbf{w}[j] + \alpha \underbrace{\sum_{i=1}^{n} \left(\left(t_i - \mathbb{M}_{\mathbf{w}} \left(\mathbf{d}_i \right) \right) \times d_i[j] \right)}_{errorDelta(\mathcal{D}, \mathbf{w}[j])}$$

Initial Weights (Iteration 2)

w[0]:	-0.146	**w**[1]:	0.233	**w**[2]:	-0.043	**w**[3]:	0.121

Exercise

$$\mathbf{w}[1] \leftarrow -0.233 + 0.00000002 \times 2195616.08 = 0.27691232$$

New Weights (Iteration 2)

w[0]:	-0.145	**w**[1]:	0.277	**w**[2]:	-0.043	**w**[3]:	0.123

- The algorithm then keeps iteratively applying the weight update rule until it converges on a stable set of weights beyond which little improvement in model accuracy is possible.
- After 100 iterations the final values for the weights are:
  - $\mathbf{w}[0] = -0.1513$,
  - $\mathbf{w}[1] = 0.6270$,
  - $\mathbf{w}[2] = -0.1781$
  - $\mathbf{w}[3] = 0.0714$
- which results in a sum of squared errors value of $2,913.5$