# Practical Lecture 3 - Learning by Optimization

Luis Sa-Couto[1] and Andreas Wichert[2]

INESC-ID, Instituto Superior Tecnico, Universidade de Lisboa
{luis.sa.couto,andreas.wichert}@tecnico.ulisboa.pt

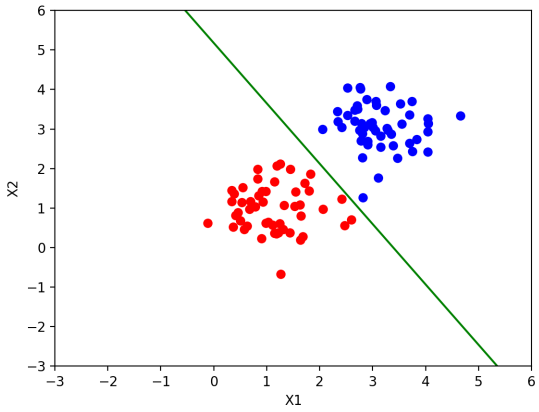
## 1 Closed form learning

### Linear Regression

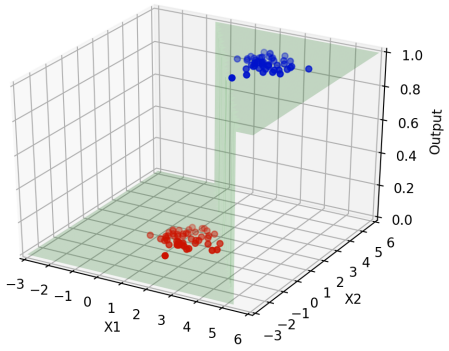
In this section we will be focusing on the model of linear regression. Much like perceptron, this model works by learning a weight vector $\mathbf{w} = \begin{pmatrix} w_0 & w_1 & \cdots & w_d \end{pmatrix}^T$ that is used to compute a linear combination of the input features $\mathbf{x} = \begin{pmatrix} x_1 & \cdots & x_d \end{pmatrix}^T$. However, this model focuses on regression instead of classification. So, for that reason, its output is not binary. It is, in fact, a real number $output\,(\mathbf{x}; \mathbf{w}) = o \in \mathbb{R}$. So achieve that, we get rid of the sign function and work directly with the linear combination (dot product).

We can think about the linear regression as providing a score for each point. Instead of making a binary decision for each point (i.e. belongs to the class or not) we give a score to each point where higher scores are more likely to belong to the class.

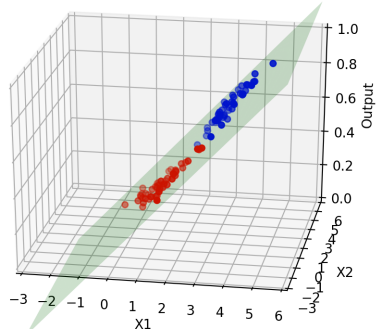
The following image shows a linearly separable problem with two input features:




If we add a third dimension for the output, we can think of the perceptron as putting all points that don't belong to the class on the $Output = 0$ plane and all points that belong to the class on the $Output = 1$ plane:




In linear regression, we have a similar picture but we don't make binary decisions, we provide an output score:




Looking at both plots we see that as the formulas suggest, the perceptron basically squashes the linear regression scores into a binary decision $perceptron\left(\mathbf{x};\mathbf{w}\right) = \Theta\left(linear\_regression\left(\mathbf{x};\mathbf{w}\right)\right)$.

### Closed form solution of squared error

So, let us assume we have training data $\left\{\left(\mathbf{x}^{(1)},t^{(1)}\right),\left(\mathbf{x}^{(2)},t^{(2)}\right),\ldots,\left(\mathbf{x}^{(n)},t^{(n)}\right)\right\}$. For a given weight vector $\mathbf{w}$, the linear regression outputs are:

$$
\begin{aligned}
output^{(1)}\left(\mathbf{x}^{(1)};\mathbf{w}\right) &= o^{(1)} = \mathbf{w}^T\mathbf{x}^{(1)} \\
output^{(2)}\left(\mathbf{x}^{(2)};\mathbf{w}\right) &= o^{(2)} = \mathbf{w}^T\mathbf{x}^{(2)} \\
&\vdots \\
output^{(n)}\left(\mathbf{x}^{(n)};\mathbf{w}\right) &= o^{(n)} = \mathbf{w}^{\mathbf{T}}\mathbf{x}^{(n)}
\end{aligned}
$$

All computations we will need to do will be made easier by working in vector notation. So, we can write the whole system of equations as follows:

$$
\begin{aligned}
\begin{pmatrix} o^{(1)} \\ o^{(2)} \\ \vdots \\ o^{(n)} \end{pmatrix} &= \begin{pmatrix} 1 & x_1^{(1)} & \cdots & x_d^{(1)} \\ 1 & x_1^{(2)} & \cdots & x_d^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(n)} & \cdots & x_d^{(n)} \end{pmatrix} \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \end{pmatrix} \\
\mathbf{O} &= \mathbf{X}\mathbf{w}
\end{aligned}
$$

Having stated how linear regression maps inputs to outputs, we can focus on the central problem of learning. We will look into the most common form of learning. That is, learning by optimization. More specifically, we want to find a weight vector such that our outputs closely approximate the targets $o^{(i)} \sim t^{(i)}$. To find that vector, it is common to define an error function that measures how wrong the outputs are. A well known error function is the mean squared error:

$$
E\left(\mathbf{w}\right) = \sum_{i=1}^{n}\left(t^{(i)} - o^{(i)}\right)^2
$$

Again, we can switch to vector notation as follows:

$$
\begin{aligned}
E\left(\mathbf{w}\right) &= \sum_{i=1}^{n}\left(t^{(i)} - o^{(i)}\right)^2 \\
&= \sum_{i=1}^{n}\left(t^{(i)} - \mathbf{w}^T\mathbf{x}^{(i)}\right)^2 \\
&= \left(\mathbf{T} - \mathbf{X}\mathbf{w}\right)^T\left(\mathbf{T} - \mathbf{X}\mathbf{w}\right)
\end{aligned}
$$

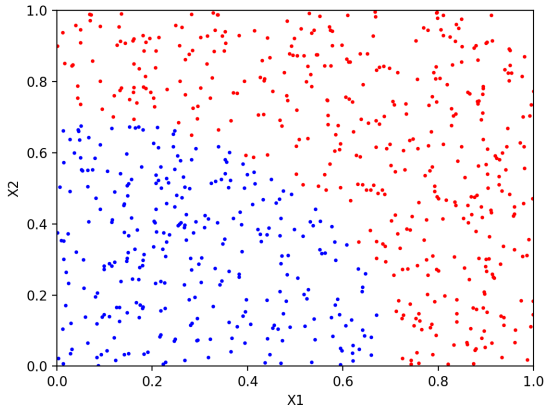
Having defined the error function, the learning problem can be solved by finding its minimum. For linear regression, we can compute a closed form solution for this problem. We do it by computing the derivative and finding its zero:

$$
\begin{aligned}
\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}} &= 0 \\
\frac{\partial (\mathbf{T}-\mathbf{X}\mathbf{w})^T (\mathbf{T}-\mathbf{X}\mathbf{w})}{\partial \mathbf{w}} &= 0 \\
\left(\frac{\partial}{\partial \mathbf{w}} (\mathbf{T}-\mathbf{X}\mathbf{w})^T\right)(\mathbf{T}-\mathbf{X}\mathbf{w}) + (\mathbf{T}-\mathbf{X}\mathbf{w})^T \left(\frac{\partial}{\partial \mathbf{w}} (\mathbf{T}-\mathbf{X}\mathbf{w})\right) &= 0 \\
\left(-\mathbf{X}^T\right)(\mathbf{T}-\mathbf{X}\mathbf{w}) + (\mathbf{T}-\mathbf{X}\mathbf{w})^T (-\mathbf{X}) &= 0 \\
\left(-\mathbf{X}^T\right)(\mathbf{T}-\mathbf{X}\mathbf{w}) + (-\mathbf{X})^T (\mathbf{T}-\mathbf{X}\mathbf{w}) &= 0 \\
-2\mathbf{X}^T (\mathbf{T}-\mathbf{X}\mathbf{w}) &= 0 \\
\mathbf{X}^T (\mathbf{T}-\mathbf{X}\mathbf{w}) &= 0 \\
\mathbf{X}^T\mathbf{T} - \mathbf{X}^T\mathbf{X}\mathbf{w} &= 0 \\
\mathbf{X}^T\mathbf{X}\mathbf{w} &= \mathbf{X}^T\mathbf{T} \\
\mathbf{w} &= \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{T}
\end{aligned}
$$

So, unlike perceptron where we had to take multiple iterative steps to learn the correct weights, in linear regression we can learn them directly.

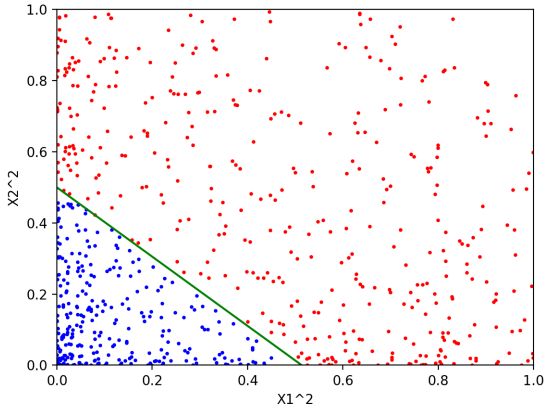
### Feature Transformations

Much like perceptron, the typical linear regression can only work well with linear problems. However, in many problems we can fix this by changing the feature space we are working in. For example, the following image shows points of two different classes that cannot be separated by a line:




However, if we notice that the difference between classes is related to their distance from the origin (point $(0, 0)$) we can work with the squares of the

features $x_1^2, x_2^2$ instead of the features themselves. The following plot shows how this transformation turns the problem into a linear one. In this space, we could use a perceptron to solve the classification issue perfectly.




This idea of feature transformation can be used in general (both in classification and regression) to transform non-linear problems into linear ones. In practice, we take the input matrix:

$$\mathbf{X} = \begin{pmatrix} 1 & x_1^{(1)} & \cdots & x_d^{(1)} \\ 1 & x_1^{(2)} & \cdots & x_d^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(n)} & \cdots & x_d^{(n)} \end{pmatrix}$$

and we choose feature transforming functions (which in the previous example correspond to taking the squares of the features $\phi_k^{(i)}\left(x_1^{(i)}, \ldots, x_d^{(i)}\right) = x_k^{(i)2}$) to build the new input matrix:

$$\mathbf{\Phi} = \begin{pmatrix} 1 & \phi_1^{(1)}\left(x_1^{(1)}, \ldots, x_d^{(1)}\right) & \cdots & \phi_l^{(1)}\left(x_1^{(1)}, \ldots, x_d^{(1)}\right) \\ 1 & \phi_1^{(2)}\left(x_1^{(2)}, \ldots, x_d^{(2)}\right) & \cdots & \phi_l^{(2)}\left(x_1^{(2)}, \ldots, x_d^{(2)}\right) \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \phi_1^{(n)}\left(x_1^{(n)}, \ldots, x_d^{(n)}\right) & \cdots & \phi_l^{(n)}\left(x_1^{(n)}, \ldots, x_d^{(n)}\right) \end{pmatrix}$$

the rest of the process stays exactly the same. We just have a different input!

**1)** Consider the following training data:

$$\left\{\mathbf{x}^{(1)} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \mathbf{x}^{(2)} = \begin{pmatrix} 2 \\ 1 \end{pmatrix}, \mathbf{x}^{(3)} = \begin{pmatrix} 1 \\ 3 \end{pmatrix}, \mathbf{x}^{(4)} = \begin{pmatrix} 3 \\ 3 \end{pmatrix}\right\}$$

$$\left\{t^{(1)} = 1.4, t^{(2)} = 0.5, t^{(3)} = 2, t^{(4)} = 2.5\right\}$$

a) Find the closed form solution for a linear regression that minimizes the sum of squared errors on the training data..

---

**Solution:**

First, we need to build the $n \times (d+1)$ design matrix to account for the bias parameter, where $n$ is the number of examples and $d$ is the original number of input features.

$$X = \begin{pmatrix} 1\ 1\ 1 \\ 1\ 2\ 1 \\ 1\ 1\ 3 \\ 1\ 3\ 3 \end{pmatrix}$$

Second, we construct a target vector:

$$Y = \begin{pmatrix} 1.4 \\ 0.5 \\ 2.0 \\ 2.5 \end{pmatrix}$$

Now, the goal is to find the weight vector $\mathbf{w} = \begin{pmatrix} w_0\ w_1\ w_2 \end{pmatrix}^T$ that minimizes the sum of squared errors. We can do it using the pseudo-inverse:

$$\mathbf{w} = \left(X^T X\right)^{-1} X^T Y$$

$$
\begin{aligned}
\mathbf{w} &= \left(X^T X\right)^{-1} X^T Y \\
&= \left[\begin{pmatrix}1&1&1\\1&2&1\\1&1&3\\1&3&3\end{pmatrix}^T \begin{pmatrix}1&1&1\\1&2&1\\1&1&3\\1&3&3\end{pmatrix}\right]^{-1} \begin{pmatrix}1&1&1\\1&2&1\\1&1&3\\1&3&3\end{pmatrix}^T \begin{pmatrix}1.4\\0.5\\2.0\\2.5\end{pmatrix} \\
&= \left[\begin{pmatrix}1&1&1&1\\1&2&1&3\\1&1&3&3\end{pmatrix} \begin{pmatrix}1&1&1\\1&2&1\\1&1&3\\1&3&3\end{pmatrix}\right]^{-1} \begin{pmatrix}1&1&1&1\\1&2&1&3\\1&1&3&3\end{pmatrix} \begin{pmatrix}1.4\\0.5\\2.0\\2.5\end{pmatrix} \\
&= \begin{pmatrix}4&7&8\\7&15&15\\8&15&20\end{pmatrix}^{-1} \begin{pmatrix}1&1&1&1\\1&2&1&3\\1&1&3&3\end{pmatrix} \begin{pmatrix}1.4\\0.5\\2.0\\2.5\end{pmatrix} \\
&= \begin{pmatrix}1.875&-0.5&-0.375\\-0.5&0.4&-0.1\\-0.375&-0.1&0.275\end{pmatrix} \begin{pmatrix}1&1&1&1\\1&2&1&3\\1&1&3&3\end{pmatrix} \begin{pmatrix}1.4\\0.5\\2.0\\2.5\end{pmatrix} \\
&= \begin{pmatrix}1.0&0.5&0.25&-0.75\\-0.2&0.2&-0.4&0.4\\-0.2&-0.3&0.35&0.15\end{pmatrix} \begin{pmatrix}1.4\\0.5\\2.0\\2.5\end{pmatrix} \\
&= \begin{pmatrix}0.275\\0.02\\0.645\end{pmatrix}
\end{aligned}
$$

---

b) Predict the target value for $x_{query} = \begin{pmatrix}2&3\end{pmatrix}^T$.

---

**Solution:**
From the previous question, we have our weights:

$$\mathbf{w} = \begin{pmatrix}0.275\\0.02\\0.645\end{pmatrix}$$

So, to compute the predicted value, we just need to augment the query vector with a bias dimension and apply the linear regression:

$$output\left(\mathbf{x}\right) = \mathbf{w} \cdot \mathbf{x} = \begin{pmatrix}0.275\\0.02\\0.645\end{pmatrix} \cdot \begin{pmatrix}1\\2\\3\end{pmatrix} = 2.25$$

---

c) Sketch the predicted hyperplane along which the linear regression predicts points will fall.

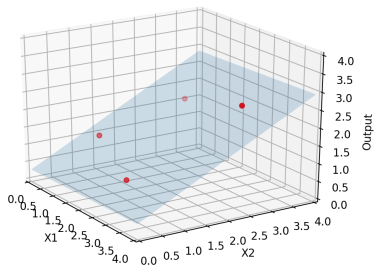
---

**Solution:**

We can get the hyperplane's equation by taking the linear regression output for a general input $\begin{pmatrix} 1 & x_1 & x_2 \end{pmatrix}^T$ and equating it to zero:

$$\begin{aligned} \textit{output}\,(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} &= 0 \\ = \begin{pmatrix} 0.275 \\ 0.02 \\ 0.645 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ x_1 \\ x_2 \end{pmatrix} &= 0 \\ = 0.02x_1 + 0.645x_1 + 0.275 &= 0 \end{aligned}$$

From the equation, we get the following plot:




---

d) Compute the mean squared error produced by the the linear regression.

---

**Solution:**

For each point in the training data, we must compute the linear regression prediction and then compute its squared error:

$$\left(t^{(1)} - \textit{output}\left(\mathbf{x}^{(1)}\right)\right)^2 = \left(t^{(1)} - \mathbf{w} \cdot \mathbf{x}^{(1)}\right)^2 = \left(1.4 - \begin{pmatrix} 0.275 \\ 0.02 \\ 0.645 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}\right)^2 = (1.4 - 0.94)^2 = 0.2116$$

$$\left(t^{(2)} - \textit{output}\left(\mathbf{x}^{(2)}\right)\right)^2 = \left(t^{(2)} - \mathbf{w} \cdot \mathbf{x}^{(2)}\right)^2 = \left(0.5 - \begin{pmatrix} 0.275 \\ 0.02 \\ 0.645 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix}\right)^2 = (0.5 - 0.96)^2 = 0.2116$$

$$\left(t^{(3)} - \textit{output}\left(\mathbf{x}^{(3)}\right)\right)^2 = \left(t^{(3)} - \mathbf{w} \cdot \mathbf{x}^{(3)}\right)^2 = \left(2.0 - \begin{pmatrix} 0.275 \\ 0.02 \\ 0.645 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \\ 3 \end{pmatrix}\right)^2 = (2.0 - 2.23)^2 = 0.0529$$

$$\left(t^{(4)} - output\left(\mathbf{x}^{(4)}\right)\right)^2 = \left(t^{(4)} - \mathbf{w}\cdot\mathbf{x}^{(4)}\right)^2 = \left(2.5 - \begin{pmatrix}0.275\\0.02\\0.645\end{pmatrix}\cdot\begin{pmatrix}1\\3\\3\end{pmatrix}\right)^2 = (2.5-2.27)^2 = 0.0529$$

So, the mean error is:

$$\frac{0.2116 + 0.2116 + 0.0529 + 0.0529}{4} = 0.13225$$

---

**2)** Consider the following training data:

$$\left\{\mathbf{x}^{(1)} = (-2.0), \mathbf{x}^{(2)} = (-1.0), \mathbf{x}^{(3)} = (0.0), \mathbf{x}^{(4)} = (2.0)\right\}$$

$$\left\{t^{(1)} = 2.0, t^{(2)} = 3.0, t^{(3)} = 1.0, t^{(4)} = -1.0\right\}$$

a) Find the closed form solution for a linear regression that minimizes the sum of squared errors on the training data..

---

**Solution:**

First, we need to build the $n \times (d+1)$ design matrix to account for the bias parameter, where $n$ is the number of examples and $d$ is the original number of input features.

$$X = \begin{pmatrix}1 & -2.0\\1 & -1.0\\1 & 0.0\\1 & 2.0\end{pmatrix}$$

Second, we construct a target vector:

$$Y = \begin{pmatrix}2.0\\3.0\\1.0\\-1.0\end{pmatrix}$$

Now, the goal is to find the weight vector $\mathbf{w} = \begin{pmatrix}w_0 & w_1\end{pmatrix}^T$ that minimizes the sum of squared errors. We can do it using the pseudo-inverse:

$$\mathbf{w} = \left(X^T X\right)^{-1} X^T Y$$

$$
\begin{aligned}
\mathbf{w} &= \left(X^T X\right)^{-1} X^T Y \\
&= \left[ \begin{pmatrix} 1 & -2.0 \\ 1 & -1.0 \\ 1 & 0.0 \\ 1 & 2.0 \end{pmatrix}^T \begin{pmatrix} 1 & -2.0 \\ 1 & -1.0 \\ 1 & 0.0 \\ 1 & 2.0 \end{pmatrix} \right]^{-1} \begin{pmatrix} 1 & -2.0 \\ 1 & -1.0 \\ 1 & 0.0 \\ 1 & 2.0 \end{pmatrix}^T \begin{pmatrix} 2.0 \\ 3.0 \\ 1.0 \\ -1.0 \end{pmatrix} \\
&= \left[ \begin{pmatrix} 1 & 1 & 1 & 1 \\ -2.0 & -1.0 & 0.0 & 2.0 \end{pmatrix} \begin{pmatrix} 1 & -2.0 \\ 1 & -1.0 \\ 1 & 0.0 \\ 1 & 2.0 \end{pmatrix} \right]^{-1} \begin{pmatrix} 1 & 1 & 1 & 1 \\ -2.0 & -1.0 & 0.0 & 2.0 \end{pmatrix} \begin{pmatrix} 2.0 \\ 3.0 \\ 1.0 \\ -1.0 \end{pmatrix} \\
&= \begin{pmatrix} 4.0 & -1.0 \\ -1.0 & 9.0 \end{pmatrix}^{-1} \begin{pmatrix} 1 & 1 & 1 & 1 \\ -2.0 & -1.0 & 0.0 & 2.0 \end{pmatrix} \begin{pmatrix} 2.0 \\ 3.0 \\ 1.0 \\ -1.0 \end{pmatrix} \\
&= \begin{pmatrix} 0.2571 & 0.0286 \\ 0.0286 & 0.1143 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 & 1 \\ -2.0 & -1.0 & 0.0 & 2.0 \end{pmatrix} \begin{pmatrix} 2.0 \\ 3.0 \\ 1.0 \\ -1.0 \end{pmatrix} \\
&= \begin{pmatrix} 0.2 & 0.2286 & 0.2571 & 0.3143 \\ -0.2 & -0.0857 & 0.0286 & 0.2571 \end{pmatrix} \begin{pmatrix} 2.0 \\ 3.0 \\ 1.0 \\ -1.0 \end{pmatrix} \\
&= \begin{pmatrix} 1.0286 \\ -0.8857 \end{pmatrix}
\end{aligned}
$$

---

b) Predict the target value for $x_{query} = (1)^T$.

---

**Solution:**
From the previous question, we have our weights:

$$\mathbf{w} = \begin{pmatrix} 1.0286 \\ -0.8857 \end{pmatrix}$$

So, to compute the predicted value, we just need to augment the query vector with a bias dimension and apply the linear regression:

$$output\,(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} = \begin{pmatrix} 1.0286 \\ -0.8857 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \end{pmatrix} = 0.1429$$

---

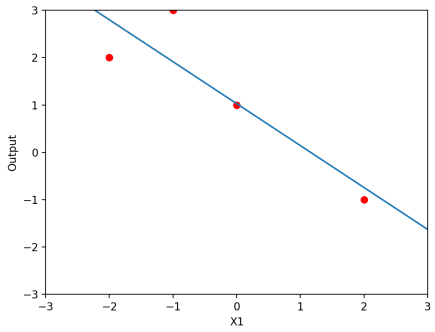
c) Sketch the predicted hyperplane along which the linear regression predicts points will fall.

**Solution:**

We can get the hyperplane's equation by taking the linear regression output for a general input $\begin{pmatrix} 1 & x_1 \end{pmatrix}^T$ and equating it to zero:

$$\begin{aligned} \textit{output}\,(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} &= 0 \\ = \begin{pmatrix} 1.0286 \\ -0.8857 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ x_1 \end{pmatrix} &= 0 \\ = -0.8857 x_1 + 1.0286 &= 0 \end{aligned}$$

From the equation, we get the following plot:




d) Compute the mean squared error produced by the the linear regression.

**Solution:**

For each point in the training data, we must compute the linear regression prediction and then compute its squared error:

$$\left(t^{(1)} - \mathbf{w} \cdot \mathbf{x}^{(1)}\right)^2 = \left(2.0 - \begin{pmatrix} 1.0286 \\ -0.8857 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ -2.0 \end{pmatrix}\right)^2 = (2.0 - 2.800)^2 = 0.64$$

$$\left(t^{(2)} - \mathbf{w} \cdot \mathbf{x}^{(2)}\right)^2 = \left(3.0 - \begin{pmatrix} 1.0286 \\ -0.8857 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ -1.0 \end{pmatrix}\right)^2 = (3.0 - 1.9143)^2 = 1.1788$$

$$\left(t^{(3)} - \mathbf{w} \cdot \mathbf{x}^{(3)}\right)^2 = \left(1.0 - \begin{pmatrix} 1.0286 \\ -0.8857 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 0.0 \end{pmatrix}\right)^2 = (1.0 - 1.0286)^2 = 0.0008$$

$$\left(t^{(4)} - \mathbf{w} \cdot \mathbf{x}^{(4)}\right)^2 = \left(-1.0 - \begin{pmatrix} 1.0286 \\ -0.8857 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 2.0 \end{pmatrix}\right)^2 = (-1.0 - (-0.7429))^2 = 0.0661$$

So, the mean error is:

$$\frac{0.64 + 1.1788 + 0.0008 + 0.0661}{4} = 0.4714$$

---

**3)** Consider the following training data:

$$\left\{\mathbf{x}^{(1)} = \begin{pmatrix}1\\1\end{pmatrix}, \mathbf{x}^{(2)} = \begin{pmatrix}2\\1\end{pmatrix}, \mathbf{x}^{(3)} = \begin{pmatrix}1\\3\end{pmatrix}, \mathbf{x}^{(4)} = \begin{pmatrix}3\\3\end{pmatrix}\right\}$$

$$\left\{t^{(1)} = 1, t^{(2)} = 1, t^{(3)} = 0, t^{(4)} = 0\right\}$$

a) Find the closed form solution for a linear regression that minimizes the sum of squared errors on the training data..

---

**Solution:**

First, we need to build the $n \times (d+1)$ design matrix to account for the bias parameter, where $n$ is the number of examples and $d$ is the original number of input features.

$$X = \begin{pmatrix}1\ 1\ 1\\1\ 2\ 1\\1\ 1\ 3\\1\ 3\ 3\end{pmatrix}$$

Second, we construct a target vector:

$$Y = \begin{pmatrix}1\\1\\0\\0\end{pmatrix}$$

Now, the goal is to find the weight vector $\mathbf{w} = \begin{pmatrix}w_0\ w_1\ w_2\end{pmatrix}^T$ that minimizes the sum of squared errors. We can do it using the pseudo-inverse:

$$\mathbf{w} = \left(X^T X\right)^{-1} X^T Y$$

$$
\begin{aligned}
\mathbf{w} &= \left(X^T X\right)^{-1} X^T Y \\
&= \left[\begin{pmatrix}1&1&1\\1&2&1\\1&1&3\\1&3&3\end{pmatrix}^T \begin{pmatrix}1&1&1\\1&2&1\\1&1&3\\1&3&3\end{pmatrix}\right]^{-1} \begin{pmatrix}1&1&1\\1&2&1\\1&1&3\\1&3&3\end{pmatrix}^T \begin{pmatrix}1\\1\\0\\0\end{pmatrix} \\
&= \left[\begin{pmatrix}1&1&1&1\\1&2&1&3\\1&1&3&3\end{pmatrix} \begin{pmatrix}1&1&1\\1&2&1\\1&1&3\\1&3&3\end{pmatrix}\right]^{-1} \begin{pmatrix}1&1&1&1\\1&2&1&3\\1&1&3&3\end{pmatrix} \begin{pmatrix}1\\1\\0\\0\end{pmatrix} \\
&= \begin{pmatrix}4&7&8\\7&15&15\\8&15&20\end{pmatrix}^{-1} \begin{pmatrix}1&1&1&1\\1&2&1&3\\1&1&3&3\end{pmatrix} \begin{pmatrix}1\\1\\0\\0\end{pmatrix} \\
&= \begin{pmatrix}1.875&-0.5&-0.375\\-0.5&0.4&-0.1\\-0.375&-0.1&0.275\end{pmatrix} \begin{pmatrix}1&1&1&1\\1&2&1&3\\1&1&3&3\end{pmatrix} \begin{pmatrix}1\\1\\0\\0\end{pmatrix} \\
&= \begin{pmatrix}1.0&0.5&0.25&-0.75\\-0.2&0.2&-0.4&0.4\\-0.2&-0.3&0.35&0.15\end{pmatrix} \begin{pmatrix}1\\1\\0\\0\end{pmatrix} \\
&= \begin{pmatrix}1.5\\0.0\\-0.5\end{pmatrix}
\end{aligned}
$$

---

b) Use your linear regression to classify $x_{query} = \begin{pmatrix}2&2.5\end{pmatrix}^T$, assuming a threshold similarity of 0.5.

---

**Solution:**
From the previous question, we have our weights:

$$\mathbf{w} = \begin{pmatrix}1.5\\0.0\\-0.5\end{pmatrix}$$

So, to compute the predicted value, we just need to augment the query vector with a bias dimension and apply the linear regression:

$$output\left(\mathbf{x}_{\text{query}}\right) = \mathbf{w} \cdot x_{\text{query}} = \begin{pmatrix}1.5\\0.0\\-0.5\end{pmatrix} \cdot \begin{pmatrix}1\\2\\2.5\end{pmatrix} = 0.25$$

The input is classified as not belonging to the class.

---

**4)** Consider the following training data:

$$\left\{\mathbf{x}^{(1)} = (-2.0)\,, \mathbf{x}^{(2)} = (-1.0)\,, \mathbf{x}^{(3)} = (0.0)\,, \mathbf{x}^{(4)} = (2.0)\right\}$$

$$\left\{t^{(1)} = 1, t^{(2)} = 0, t^{(3)} = 0, t^{(4)} = 0\right\}$$

a) Find the closed form solution for a linear regression that minimizes the sum of squared errors on the training data..

---

**Solution:**

First, we need to build the $n \times (d+1)$ design matrix to account for the bias parameter, where $n$ is the number of examples and $d$ is the original number of input features.

$$X = \begin{pmatrix} 1 & -2.0 \\ 1 & -1.0 \\ 1 & 0.0 \\ 1 & 2.0 \end{pmatrix}$$

Second, we construct a target vector:

$$Y = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Now, the goal is to find the weight vector $\mathbf{w} = \begin{pmatrix} w_0 & w_1 \end{pmatrix}^T$ that minimizes the sum of squared errors. We can do it using the pseudo-inverse:

$$\mathbf{w} = \left(X^T X\right)^{-1} X^T Y$$

$$
\begin{aligned}
\mathbf{w} &= \left(X^T X\right)^{-1} X^T Y \\
&= \left[\begin{pmatrix} 1 & -2.0 \\ 1 & -1.0 \\ 1 & 0.0 \\ 1 & 2.0 \end{pmatrix}^T \begin{pmatrix} 1 & -2.0 \\ 1 & -1.0 \\ 1 & 0.0 \\ 1 & 2.0 \end{pmatrix}\right]^{-1} \begin{pmatrix} 1 & -2.0 \\ 1 & -1.0 \\ 1 & 0.0 \\ 1 & 2.0 \end{pmatrix}^T \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \\
&= \left[\begin{pmatrix} 1 & 1 & 1 & 1 \\ -2.0 & -1.0 & 0.0 & 2.0 \end{pmatrix} \begin{pmatrix} 1 & -2.0 \\ 1 & -1.0 \\ 1 & 0.0 \\ 1 & 2.0 \end{pmatrix}\right]^{-1} \begin{pmatrix} 1 & 1 & 1 & 1 \\ -2.0 & -1.0 & 0.0 & 2.0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \\
&= \begin{pmatrix} 4.0 & -1.0 \\ -1.0 & 9.0 \end{pmatrix}^{-1} \begin{pmatrix} 1 & 1 & 1 & 1 \\ -2.0 & -1.0 & 0.0 & 2.0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \\
&= \begin{pmatrix} 0.2571 & 0.0286 \\ 0.0286 & 0.1143 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 & 1 \\ -2.0 & -1.0 & 0.0 & 2.0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \\
&= \begin{pmatrix} 0.2 & 0.2286 & 0.2571 & 0.3143 \\ -0.2 & -0.0857 & 0.0286 & 0.2571 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \\
&= \begin{pmatrix} 0.2 \\ -0.2 \end{pmatrix}
\end{aligned}
$$

---

b) Use your linear regression to classify $x_{query} = (-0.3)^T$, assuming a threshold similarity of 0.15.

---

**Solution:**
From the previous question, we have our weights:

$$\mathbf{w} = \begin{pmatrix} 0.2 \\ -0.2 \end{pmatrix}$$

So, to compute the predicted value, we just need to augment the query vector with a bias dimension and apply the linear regression:

$$output\left(\mathbf{x}_{\text{query}}\right) = \mathbf{w} \cdot \mathbf{x}_{\text{query}} = \begin{pmatrix} 0.2 \\ -0.2 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ -0.3 \end{pmatrix} = 0.26$$

The input is classified as belonging to the class.

---

**5)** Consider the following training data:

$$\mathbf{x}^{(1)} = \begin{pmatrix} -0.95 \\ 0.62 \end{pmatrix}, \mathbf{x}^{(2)} = \begin{pmatrix} 0.63 \\ 0.31 \end{pmatrix}, \mathbf{x}^{(3)} = \begin{pmatrix} -0.12 \\ -0.21 \end{pmatrix}, \mathbf{x}^{(4)} = \begin{pmatrix} -0.24 \\ -0.5 \end{pmatrix},$$
$$\mathbf{x}^{(5)} = \begin{pmatrix} 0.07 \\ -0.42 \end{pmatrix}, \mathbf{x}^{(6)} = \begin{pmatrix} 0.03 \\ 0.91 \end{pmatrix}, \mathbf{x}^{(7)} = \begin{pmatrix} 0.05 \\ 0.09 \end{pmatrix}, \mathbf{x}^{(8)} = \begin{pmatrix} -0.83 \\ 0.22 \end{pmatrix}$$

$$\left\{ t^{(1)} = 0, t^{(2)} = 0, t^{(3)} = 1, t^{(4)} = 0, t^{(5)} = 1, t^{(6)} = 0, t^{(7)} = 1, t^{(8)} = 0 \right\}$$

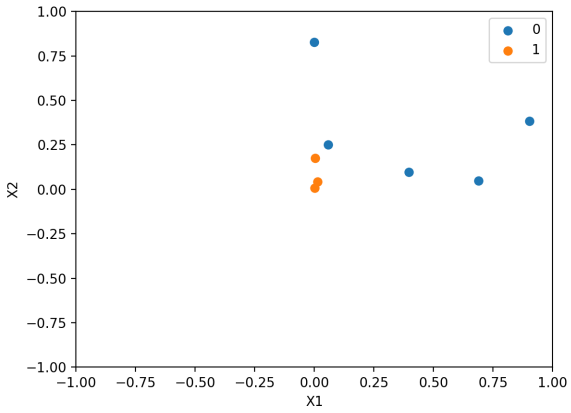
a) Plot the data points and try to choose a non-linear transformation to apply.

---

**Solution:**

Plotting the data points we see that the labels seem to change with the distance from the origin. A way to capture is this is to perform a quadratic feature transform:

$$\phi\left(x_1, x_2\right) = \left(x_1^2, x_2^2\right)$$




---

b) Adopt the non-linear transform you chose in a) and find the closed form solution.

---

**Solution:**

First, we need to build the $n \times (d+1)$ design matrix to account for the bias parameter, where $n$ is the number of examples and $d$ is the original number of input features. However, unlike previous exercises where we used the features themselves directly ($\phi\left(x\right) = x$), in this case, we have a non-linear transformation. So, we apply it:

$$\Phi = \begin{pmatrix} 1 & (-0.95)^2 & (0.62)^2 \\ 1 & (0.63)^2 & (0.31)^2 \\ 1 & (-0.12)^2 & (-0.21)^2 \\ 1 & (-0.24)^2 & (-0.5)^2 \\ 1 & (0.07)^2 & (-0.42)^2 \\ 1 & (0.03)^2 & (0.91)^2 \\ 1 & (0.05)^2 & (0.09)^2 \\ 1 & (-0.83)^2 & (0.22)^2 \end{pmatrix} = \begin{pmatrix} 1 & 0.9025 & 0.3844 \\ 1 & 0.3969 & 0.0961 \\ 1 & 0.0144 & 0.0441 \\ 1 & 0.0576 & 0.2500 \\ 1 & 0.0049 & 0.1764 \\ 1 & 0.0009 & 0.8281 \\ 1 & 0.0025 & 0.0081 \\ 1 & 0.6889 & 0.0484 \end{pmatrix}$$

Second, we construct a target vector:

$$Y = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

Now, the goal is to find the weight vector $\mathbf{w} = \begin{pmatrix} w_0 & w_1 & w_2 \end{pmatrix}^T$ that minimizes the sum of squared errors. We can do it using the pseudo-inverse:

$$\mathbf{w} = \left(X^T X\right)^{-1} X^T Y$$

$$\mathbf{w} = \left(X^T X\right)^{-1} X^T Y$$

$$= \left[ \begin{pmatrix} 1 & 0.9025 & 0.3844 \\ 1 & 0.3969 & 0.0961 \\ 1 & 0.0144 & 0.0441 \\ 1 & 0.0576 & 0.2500 \\ 1 & 0.0049 & 0.1764 \\ 1 & 0.0009 & 0.8281 \\ 1 & 0.0025 & 0.0081 \\ 1 & 0.6889 & 0.0484 \end{pmatrix}^T \begin{pmatrix} 1 & 0.9025 & 0.3844 \\ 1 & 0.3969 & 0.0961 \\ 1 & 0.0144 & 0.0441 \\ 1 & 0.0576 & 0.2500 \\ 1 & 0.0049 & 0.1764 \\ 1 & 0.0009 & 0.8281 \\ 1 & 0.0025 & 0.0081 \\ 1 & 0.6889 & 0.0484 \end{pmatrix} \right]^{-1} \begin{pmatrix} 1 & 0.9025 & 0.3844 \\ 1 & 0.3969 & 0.0961 \\ 1 & 0.0144 & 0.0441 \\ 1 & 0.0576 & 0.2500 \\ 1 & 0.0049 & 0.1764 \\ 1 & 0.0009 & 0.8281 \\ 1 & 0.0025 & 0.0081 \\ 1 & 0.6889 & 0.0484 \end{pmatrix}^T \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

$$= \begin{pmatrix} 8.0000 & 2.0686 & 1.8356 \\ 2.0686 & 1.4502 & 0.4351 \\ 1.8356 & 0.4351 & 0.9407 \end{pmatrix}^{-1} \begin{pmatrix} 1 & 0.9025 & 0.3844 \\ 1 & 0.3969 & 0.0961 \\ 1 & 0.0144 & 0.0441 \\ 1 & 0.0576 & 0.2500 \\ 1 & 0.0049 & 0.1764 \\ 1 & 0.0009 & 0.8281 \\ 1 & 0.0025 & 0.0081 \\ 1 & 0.6889 & 0.0484 \end{pmatrix}^T \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

$$= \begin{pmatrix} 0.3099 & -0.3026 & -0.4647 \\ -0.3026 & 1.0962 & 0.0835 \\ -0.4647 & 0.0835 & 1.9311 \end{pmatrix} \begin{pmatrix} 1 & 0.9025 & 0.3844 \\ 1 & 0.3969 & 0.0961 \\ 1 & 0.0144 & 0.0441 \\ 1 & 0.0576 & 0.2500 \\ 1 & 0.0049 & 0.1764 \\ 1 & 0.0009 & 0.8281 \\ 1 & 0.0025 & 0.0081 \\ 1 & 0.6889 & 0.0484 \end{pmatrix}^T \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

$$= \begin{pmatrix} -0.1419 & 0.1451 & 0.2850 & 0.1763 & 0.2264 & -0.0752 & 0.3053 & 0.0789 \\ 0.7188 & 0.1405 & -0.2831 & -0.2186 & -0.2825 & -0.2325 & -0.2992 & 0.4566 \\ 0.3529 & -0.2459 & -0.3783 & 0.0229 & -0.1236 & 1.1346 & -0.4488 & -0.3137 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

$$= \begin{pmatrix} 0.8168 \\ -0.8648 \\ -0.9508 \end{pmatrix}$$

---

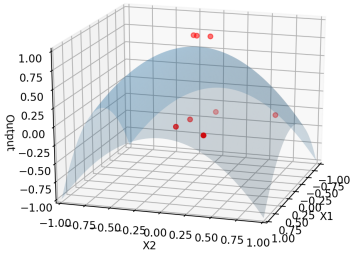
c) Sketch the predicted surface along which the predictions will fall.

---

**Solution:**

$$
\begin{aligned}
output\,(\mathbf{x}) = \mathbf{w} \cdot \phi\,(\mathbf{x}) &= 0 \\
= \begin{pmatrix} 0.8168 \\ -0.8648 \\ -0.9508 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ x_1^2 \\ x_2^2 \end{pmatrix} &= 0 \\
= -0.8648x_1^2 - 0.9508x_2^2 + 0.8168 &= 0
\end{aligned}
$$

From the equation, we get the following plot:




---

**6)** Consider the following training data:

$$\left\{\mathbf{x}^{(1)} = (3)\,, \mathbf{x}^{(2)} = (4)\,, \mathbf{x}^{(3)} = (6)\,, \mathbf{x}^{(4)} = (10)\,, \mathbf{x}^{(5)} = (12)\right\}$$

$$\left\{t^{(1)} = 1.5, t^{(2)} = 9.3, t^{(3)} = 23.4, t^{(4)} = 45.8, t^{(5)} = 60.1\right\}$$

a) Adopt a logarithmic feature transformation $\phi\,(x_1) = \log\,(x_1)$ and find the closed form solution for this non-linear regression that minimizes the sum of squared errors on the training data.

---

**Solution:**

First, we need to build the $n \times (d+1)$ design matrix to account for the bias parameter, where $n$ is the number of examples and $d$ is the original number of input features. However, unlike previous exercises where we used the features themselves directly ($\phi\,(x) = x$), in this case, we have a non-linear transformation. So, we apply it:

$$\Phi = \begin{pmatrix} 1 & \log\,(3) \\ 1 & \log\,(4) \\ 1 & \log\,(6) \\ 1 & \log\,(10) \\ 1 & \log\,(12) \end{pmatrix} = \begin{pmatrix} 1 & 1.0986 \\ 1 & 1.3863 \\ 1 & 1.7918 \\ 1 & 2.3026 \\ 1 & 2.4849 \end{pmatrix}$$

Second, we construct a target vector:

$$Y = \begin{pmatrix} 1.5 \\ 9.3 \\ 23.4 \\ 45.8 \\ 60.1 \end{pmatrix}$$

Now, the goal is to find the weight vector $\mathbf{w} = \begin{pmatrix} w_0 & w_1 \end{pmatrix}^T$ that minimizes the sum of squared errors. We can do it using the pseudo-inverse:

$$\mathbf{w} = \left(X^T X\right)^{-1} X^T Y$$

$$\begin{aligned}
\mathbf{w} &= \left(X^T X\right)^{-1} X^T Y \\
&= \left[ \begin{pmatrix} 1 & 1.0986 \\ 1 & 1.3863 \\ 1 & 1.7918 \\ 1 & 2.3026 \\ 1 & 2.4849 \end{pmatrix}^T \begin{pmatrix} 1 & 1.0986 \\ 1 & 1.3863 \\ 1 & 1.7918 \\ 1 & 2.3026 \\ 1 & 2.4849 \end{pmatrix} \right]^{-1} \begin{pmatrix} 1 & 1.0986 \\ 1 & 1.3863 \\ 1 & 1.7918 \\ 1 & 2.3026 \\ 1 & 2.4849 \end{pmatrix}^T \begin{pmatrix} 1.5 \\ 9.3 \\ 23.4 \\ 45.8 \\ 60.1 \end{pmatrix} \\
&= \left[ \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1.0986 & 1.3863 & 1.7918 & 2.3026 & 2.4849 \end{pmatrix} \begin{pmatrix} 1 & 1.0986 \\ 1 & 1.3863 \\ 1 & 1.7918 \\ 1 & 2.3026 \\ 1 & 2.4849 \end{pmatrix} \right]^{-1} \begin{pmatrix} 1 & 1.0986 \\ 1 & 1.3863 \\ 1 & 1.7918 \\ 1 & 2.3026 \\ 1 & 2.4849 \end{pmatrix}^T \begin{pmatrix} 1.5 \\ 9.3 \\ 23.4 \\ 45.8 \\ 60.1 \end{pmatrix} \\
&= \begin{pmatrix} 5.0 & 9.0642 \\ 9.0642 & 17.8158 \end{pmatrix}^{-1} \begin{pmatrix} 1 & 1.0986 \\ 1 & 1.3863 \\ 1 & 1.7918 \\ 1 & 2.3026 \\ 1 & 2.4849 \end{pmatrix}^T \begin{pmatrix} 1.5 \\ 9.3 \\ 23.4 \\ 45.8 \\ 60.1 \end{pmatrix} \\
&= \begin{pmatrix} 2.5745 & -1.3098 \\ -1.3098 & 0.7225 \end{pmatrix} \begin{pmatrix} 1 & 1.0986 \\ 1 & 1.3863 \\ 1 & 1.7918 \\ 1 & 2.3026 \\ 1 & 2.4849 \end{pmatrix}^T \begin{pmatrix} 1.5 \\ 9.3 \\ 23.4 \\ 45.8 \\ 60.1 \end{pmatrix} \\
&= \begin{pmatrix} 1.1355 & 0.7587 & 0.2276 & -0.4415 & -0.6803 \\ -0.5160 & -0.3082 & -0.0152 & 0.3539 & 0.4856 \end{pmatrix} \begin{pmatrix} 1.5 \\ 9.3 \\ 23.4 \\ 45.8 \\ 60.1 \end{pmatrix} \\
&= \begin{pmatrix} -47.0212 \\ 41.3945 \end{pmatrix}
\end{aligned}$$

b) Repeat the exercise above for a quadratic feature transformation $\phi(x_1) = x_1^2$.

---

**Solution:**

First, we need to build the $n \times (d+1)$ design matrix to account for the bias parameter, where $n$ is the number of examples and $d$ is the original number of input features. However, unlike previous exercises where we used the features themselves directly ($\phi(x) = x$), in this case, we have a non-linear transformation. So, we apply it:

$$\Phi = \begin{pmatrix} 1 & 3^2 \\ 1 & 4^2 \\ 1 & 6^2 \\ 1 & 10^2 \\ 1 & 12^2 \end{pmatrix} = \begin{pmatrix} 1 & 9 \\ 1 & 16 \\ 1 & 36 \\ 1 & 100 \\ 1 & 144 \end{pmatrix}$$

Second, we construct a target vector:

$$Y = \begin{pmatrix} 1.5 \\ 9.3 \\ 23.4 \\ 45.8 \\ 60.1 \end{pmatrix}$$

Now, the goal is to find the weight vector $\mathbf{w} = \begin{pmatrix} w_0 & w_1 \end{pmatrix}^T$ that minimizes the sum of squared errors. We can do it using the pseudo-inverse:

$$\mathbf{w} = \left(X^T X\right)^{-1} X^T Y$$

$$
\begin{aligned}
\mathbf{w} &= \left(X^T X\right)^{-1} X^T Y \\
&= \left[\begin{pmatrix} 1 & 9 \\ 1 & 16 \\ 1 & 36 \\ 1 & 100 \\ 1 & 144 \end{pmatrix}^T \begin{pmatrix} 1 & 9 \\ 1 & 16 \\ 1 & 36 \\ 1 & 100 \\ 1 & 144 \end{pmatrix}\right]^{-1} \begin{pmatrix} 1 & 9 \\ 1 & 16 \\ 1 & 36 \\ 1 & 100 \\ 1 & 144 \end{pmatrix}^T \begin{pmatrix} 1.5 \\ 9.3 \\ 23.4 \\ 45.8 \\ 60.1 \end{pmatrix} \\
&= \left[\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 9 & 16 & 36 & 100 & 144 \end{pmatrix} \begin{pmatrix} 1 & 9 \\ 1 & 16 \\ 1 & 36 \\ 1 & 100 \\ 1 & 144 \end{pmatrix}\right]^{-1} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 9 & 16 & 36 & 100 & 144 \end{pmatrix} \begin{pmatrix} 1.5 \\ 9.3 \\ 23.4 \\ 45.8 \\ 60.1 \end{pmatrix} \\
&= \begin{pmatrix} 5.0 & 305.0 \\ 305.0 & 32369.0 \end{pmatrix}^{-1} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 9 & 16 & 36 & 100 & 144 \end{pmatrix} \begin{pmatrix} 1.5 \\ 9.3 \\ 23.4 \\ 45.8 \\ 60.1 \end{pmatrix} \\
&= \begin{pmatrix} 0.4703 & -0.0044 \\ -0.0044 & 0.00007 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 9 & 16 & 36 & 100 & 144 \end{pmatrix} \begin{pmatrix} 1.5 \\ 9.3 \\ 23.4 \\ 45.8 \\ 60.1 \end{pmatrix} \\
&= \begin{pmatrix} 0.4305 & 0.3994 & 0.3108 & 0.0272 & -0.1678 \\ -0.0038 & -0.0033 & -0.0018 & 0.0028 & 0.0060 \end{pmatrix} \begin{pmatrix} 1.5 \\ 9.3 \\ 23.4 \\ 45.8 \\ 60.1 \end{pmatrix} \\
&= \begin{pmatrix} 2.7895 \\ 0.4136 \end{pmatrix}
\end{aligned}
$$

---

c) Plot both regressions.

---

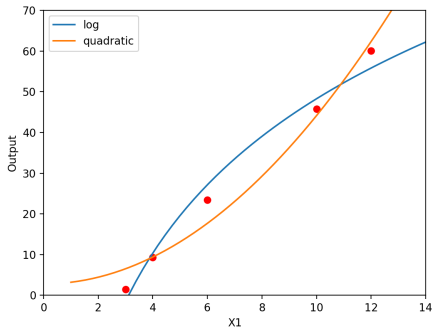
**Solution:**
For the logarithmic, we get:

$$output\left(\mathbf{x}\right) = w \cdot \phi\left(\mathbf{x}\right) = -47.0212 + 41.395 \log\left(x_1\right)$$

For the quadratic, we get:

$$output\left(\mathbf{x}\right) = w \cdot \phi\left(\mathbf{x}\right) = 2.7895 + 0.4136 x_1^2$$

With the equations, we can build the plots:




d) Which is a better fit a) or b)?

**Solution:**
Recall the equations for both regressions:

$$output_{log}\left(\mathbf{x}\right) = w \cdot \phi\left(\mathbf{x}\right) = -47.0212 + 41.395 \log\left(x_1\right)$$

$$output_{quadratic}\left(\mathbf{x}\right) = w \cdot \phi\left(\mathbf{x}\right) = 2.7895 + 0.4136 x_1^2$$

To measure which fit is better we will measure the mean squared errors.
For the logarithmic trannsform, we have:

$$\left(\mathbf{\Phi w}_{\log} - \mathbf{Y}\right)^2 = \left(\begin{pmatrix} 1 & 1.0986 \\ 1 & 1.3863 \\ 1 & 1.7918 \\ 1 & 2.3026 \\ 1 & 2.4849 \end{pmatrix} \begin{pmatrix} -47.0212 \\ 41.3945 \end{pmatrix} - \begin{pmatrix} 1.5 \\ 9.3 \\ 23.4 \\ 45.8 \\ 60.1 \end{pmatrix}\right)^2 = \begin{pmatrix} 9.2704 \\ 1.1315 \\ 14.0455 \\ 6.2155 \\ 18.1460 \end{pmatrix}$$

Which yields a mean squared error of:

$$\frac{9.2704 + 1.1315 + 14.0455 + 6.2155 + 18.1460}{5} = 9.7618$$

For the quadratic transform, we have:

$$\left(\mathbf{\Phi w}_{\text{quadratic}} - \mathbf{Y}\right)^2 = \left(\begin{pmatrix} 1 & 9 \\ 1 & 16 \\ 1 & 36 \\ 1 & 100 \\ 1 & 144 \end{pmatrix} \begin{pmatrix} 2.7895 \\ 0.4136 \end{pmatrix} - \begin{pmatrix} 1.5 \\ 9.3 \\ 23.4 \\ 45.8 \\ 60.1 \end{pmatrix}\right)^2 = \begin{pmatrix} 25.1202 \\ 0.0152 \\ 32.7228 \\ 2.7192 \\ 5.0628 \end{pmatrix}$$

Which yields a mean squared error of:

$$\frac{25.1202 + 0.0152 + 32.7228 + 2.7192 + 5.0628}{5} = 13.1273$$

So, given the availale data, the logarithmic transform appears to fit the data better.

---

## 2 Gradient descent learning

We have already saw that the perceptron can work with other activation functions besides the sign. From that knowledge it is helpful to define what is called a generalized linear unit that receives an input vector $x = \begin{pmatrix} x_0 \cdots x_d \end{pmatrix}^T$, takes the dot product with its weight vector $\mathbf{w} = \begin{pmatrix} w_0 \cdots w_d \end{pmatrix}^T$ and passes it to a general activation function $g$. We can write the output of this generalized unit as follows:

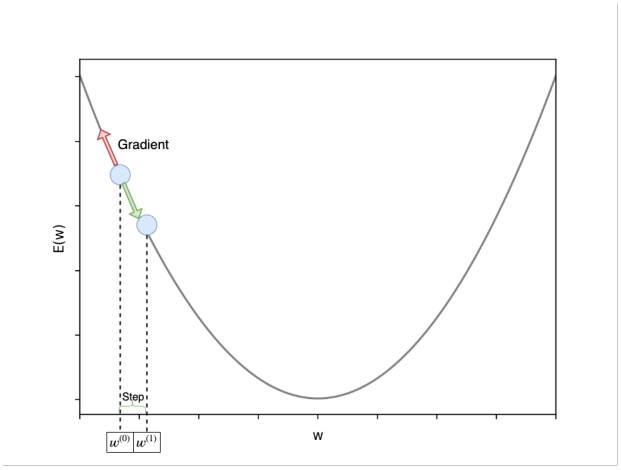
$$unit\left(\mathbf{x}; \mathbf{w}\right) = g\left(\mathbf{w}^T\mathbf{x}\right)$$

In the original perceptron, $g$ was the sign function. Afterwards, we worked with the $\Theta$ function. In fact, we can use, in principle, any function as we will se in the exercises below.

To perform learning in a generalized linear unit we also define an error function and try to minimize it. However, the process of minimization is not so easy as with linear regression. The activation function makes it impossible to get the same kind of closed form solution. So, we will have to use an iterative procedure, gradient descent!

We want to find the weight vector that minimizes an error function. However, we do not know a priori the error for all possible weight vectors. All we can do is: given an error function (like the sum of squared errors in the previous section) and training data, compute the error for a specific weight vector on that training data.

Illustrated in the figure below is the idea of gradient descent. Specifically, we start with a given weight vector $\mathbf{w}_0$ and compute the error function gradient for that vector $\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}}\left(\mathbf{w}_0\right)$. This gradient will point away from the closest minimum value of error. So, we update our weight vector by taking a step into the reverse direction of the gradient $\mathbf{w}^{(1)} = \mathbf{w}^{(0)} - \eta \frac{\partial E(\mathbf{w})}{\partial \mathbf{w}}\left(\mathbf{w}^{(0)}\right)$.




So, in general, for a differentiable error function, we have the general learning rule:

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \frac{\partial E\left(\mathbf{w}\right)}{\partial \mathbf{w}} \left(\mathbf{w}^{(t)}\right)$$

The error function gradient is computed on some training data. Sometimes, this process can take too long (if we have many examples) so we use stochastic gradient descent where the gradient of the error is computed for a single training example. So, instead of using all examples to take one update step we do one step per example. This makes learning faster. However, approximating the gradient with only one example can be very noisy so we can take very wrong steps sometimes.

Before moving to the exercises, let us make a quick note. When the error function is the cross-entropy loss and the unit's activation function is the sigmoid logistic function we call the model a logistic regression.

**1)** Consider the following training data:

$$\left\{\mathbf{x}^{(1)} = \begin{pmatrix}1\\1\end{pmatrix}, \mathbf{x}^{(2)} = \begin{pmatrix}2\\1\end{pmatrix}, \mathbf{x}^{(3)} = \begin{pmatrix}1\\3\end{pmatrix}, \mathbf{x}^{(4)} = \begin{pmatrix}3\\3\end{pmatrix}\right\}$$

$$\left\{t^{(1)} = 1, t^{(2)} = 1, t^{(3)} = 0, t^{(4)} = 0\right\}$$

In this exercise, we will work with a unit that computes the following function:

$$output\left(\mathbf{x}; \mathbf{w}\right) = \frac{1}{1 + \exp\left(-2\mathbf{w} \cdot \mathbf{x}\right)}$$

And we will use the half sum of squared errors as our error (loss) function:

$$E\left(\mathbf{w}\right) = \frac{1}{2} \sum_{k=1}^{N} \left(t^{(k)} - output\left(\mathbf{x}^{(k)}; \mathbf{w}\right)\right)^2$$

a) Determine the gradient descent learning rule for this unit.

---

**Solution:**

To apply gradient descent, we want an update rule that moves a step of size $\eta$ towards the opposite direction from the gradient of the error function with respect to the weights:

$$\mathbf{w} = \mathbf{w} - \eta \frac{\partial E(\mathbf{w})}{\partial \mathbf{w}}$$

To find the learning rule, we must compute the gradient. Before doing so, we should notice that the model is computing a sigmoid function, so:

$$output(\mathbf{x}; \mathbf{w}) = \frac{1}{1 + \exp(-2\mathbf{w} \cdot \mathbf{x})} = \sigma(2\mathbf{w} \cdot \mathbf{x})$$

Which has a well-known derivative that we will use later:

$$\begin{aligned}
\frac{\partial \sigma(x)}{\partial x} &= \frac{\partial \frac{1}{1+\exp(-x)}}{\partial x} \\
&= \frac{\partial \frac{1}{1+\exp(-x)}}{\partial (1+\exp(-x))} \frac{\partial (1+\exp(-x))}{\partial (\exp(-x))} \frac{\partial (\exp(-x))}{\partial (-x)} \frac{\partial (-x)}{\partial x} \\
&= -\frac{1}{(1+\exp(-x))^2} \exp(-x)(-1) \\
&= \frac{1}{(1+\exp(-x))^2} \exp(-x) \\
&= \frac{1}{1+\exp(-x)} \frac{\exp(-x)}{1+\exp(-x)} \\
&= \frac{1}{1+\exp(-x)} \frac{\exp(-x)+1-1}{1+\exp(-x)} \\
&= \frac{1}{1+\exp(-x)} \left( \frac{\exp(-x)+1}{1+\exp(-x)} - \frac{1}{1+\exp(-x)} \right) \\
&= \frac{1}{1+\exp(-x)} \left( 1 - \frac{1}{1+\exp(-x)} \right) \\
&= \sigma(x)(1-\sigma(x))
\end{aligned}$$

Having made this auxiliar computation, it is now easier to compute the derivative of the error function with respect to the parameter vector.

$$
\begin{aligned}
\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}} &= \frac{\partial \frac{1}{2}\sum_{k=1}^{N}\left(t^{(k)} - output\left(\mathbf{x}^{(k)};\mathbf{w}\right)\right)^2}{\partial \mathbf{w}} \\
&= \frac{\partial \frac{1}{2}\sum_{k=1}^{N}\left(t^{(k)} - \sigma\left(2\mathbf{w}\cdot\mathbf{x}^{(k)}\right)\right)^2}{\partial \mathbf{w}} \\
&= \frac{1}{2}\frac{\partial \sum_{k=1}^{N}\left(t^{(k)} - \sigma\left(2\mathbf{w}\cdot\mathbf{x}^{(k)}\right)\right)^2}{\partial \mathbf{w}} \\
&= \frac{1}{2}\sum_{k=1}^{N}\frac{\partial \left(t^{(k)} - \sigma\left(2\mathbf{w}\cdot\mathbf{x}^{(k)}\right)\right)^2}{\partial \mathbf{w}} \\
&= \frac{1}{2}\sum_{k=1}^{N}\left(\frac{\partial \left(t^{(k)} - \sigma\left(2\mathbf{w}\cdot\mathbf{x}^{(k)}\right)\right)^2}{\partial \left(t^{(k)} - \sigma\left(2\mathbf{w}\cdot\mathbf{x}^{(k)}\right)\right)}\frac{\partial \left(t^{(k)} - \sigma\left(2\mathbf{w}\cdot\mathbf{x}^{(k)}\right)\right)}{\partial \sigma\left(2\mathbf{w}\cdot\mathbf{x}^{(k)}\right)}\frac{\partial \sigma\left(2\mathbf{w}\cdot\mathbf{x}^{(k)}\right)}{\partial \left(2\mathbf{w}\cdot\mathbf{x}^{(k)}\right)}\frac{\partial \left(2\mathbf{w}\cdot\mathbf{x}^{(k)}\right)}{\partial \mathbf{w}}\right) \\
&= \frac{1}{2}\sum_{k=1}^{N}\left(2\left(t^{(k)} - \sigma\left(2\mathbf{w}\cdot\mathbf{x}^{(k)}\right)\right)\frac{\partial \left(t^{(k)} - \sigma\left(2\mathbf{w}\cdot\mathbf{x}^{(k)}\right)\right)}{\partial \sigma\left(2\mathbf{w}\cdot\mathbf{x}^{(k)}\right)}\frac{\partial \sigma\left(2\mathbf{w}\cdot\mathbf{x}^{(k)}\right)}{\partial \left(2\mathbf{w}\cdot\mathbf{x}^{(k)}\right)}\frac{\partial \left(2\mathbf{w}\cdot\mathbf{x}^{(k)}\right)}{\partial \mathbf{w}}\right) \\
&= \frac{1}{2}\sum_{k=1}^{N}\left(2\left(t^{(k)} - \sigma\left(2\mathbf{w}\cdot\mathbf{x}^{(k)}\right)\right)(-1)\frac{\partial \sigma\left(2\mathbf{w}\cdot\mathbf{x}^{(k)}\right)}{\partial \left(2\mathbf{w}\cdot\mathbf{x}^{(k)}\right)}\frac{\partial \left(2\mathbf{w}\cdot\mathbf{x}^{(k)}\right)}{\partial \mathbf{w}}\right) \\
&= \frac{1}{2}\sum_{k=1}^{N}\left(2\left(t^{(k)} - \sigma\left(2\mathbf{w}\cdot\mathbf{x}^{(k)}\right)\right)(-1)\left(\sigma\left(2\mathbf{w}\cdot\mathbf{x}^{(k)}\right)\left(1-\sigma\left(2\mathbf{w}\cdot\mathbf{x}^{(k)}\right)\right)\right)\frac{\partial \left(2\mathbf{w}\cdot\mathbf{x}^{(k)}\right)}{\partial \mathbf{w}}\right) \\
&= \frac{1}{2}\sum_{k=1}^{N}\left(2\left(t^{(k)} - \sigma\left(2\mathbf{w}\cdot\mathbf{x}^{(k)}\right)\right)(-1)\left(\sigma\left(2\mathbf{w}\cdot\mathbf{x}^{(k)}\right)\left(1-\sigma\left(2\mathbf{w}\cdot\mathbf{x}^{(k)}\right)\right)\right)\left(2\mathbf{x}^{(k)}\right)\right) \\
&= -2\sum_{k=1}^{N}\left(\left(t^{(k)} - \sigma\left(2\mathbf{w}\cdot\mathbf{x}^{(k)}\right)\right)\left(\sigma\left(2\mathbf{w}\cdot\mathbf{x}^{(k)}\right)\left(1-\sigma\left(2\mathbf{w}\cdot\mathbf{x}^{(k)}\right)\right)\right)\mathbf{x}^{(k)}\right)
\end{aligned}
$$

So, we can write our update rule as follows.

$$
\begin{aligned}
\mathbf{w} &= \mathbf{w} - \eta\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}} \\
&= \mathbf{w} - \eta\left(-2\sum_{k=1}^{N}\left(\left(t^{(k)} - \sigma\left(2\mathbf{w}\cdot\mathbf{x}^{(k)}\right)\right)\left(\sigma\left(2\mathbf{w}\cdot\mathbf{x}^{(k)}\right)\left(1-\sigma\left(2\mathbf{w}\cdot\mathbf{x}^{(k)}\right)\right)\right)\mathbf{x}^{(k)}\right)\right) \\
&= \mathbf{w} + 2\eta\sum_{k=1}^{N}\left(\left(t^{(k)} - \sigma\left(2\mathbf{w}\cdot\mathbf{x}^{(k)}\right)\right)\left(\sigma\left(2\mathbf{w}\cdot\mathbf{x}^{(k)}\right)\left(1-\sigma\left(2\mathbf{w}\cdot\mathbf{x}^{(k)}\right)\right)\right)\mathbf{x}^{(k)}\right)
\end{aligned}
$$

---

b) Compute the first gradient descent update assuming an initialization of all ones .

---

**Solution:**

The original gradient descent does one update per epoch because its learning rule requires contributions from all data points to do one step. Let us compute it.

According to the problem statement, we start with weights $\mathbf{w} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$ and learning rate $\eta = 1.0$.

$$
\begin{aligned}
\mathbf{w} &= \mathbf{w} + 2\eta \sum_{k=1}^{4} \left( \left( t^{(k)} - \sigma\left( 2\mathbf{w} \cdot \mathbf{x}^{(k)} \right) \right) \left( \sigma\left( 2\mathbf{w} \cdot \mathbf{x}^{(k)} \right) \left( 1 - \sigma\left( 2\mathbf{w} \cdot \mathbf{x}^{(k)} \right) \right) \right) \mathbf{x}^{(k)} \right) \\
&= \mathbf{w} + \sum_{k=1}^{4} 2\eta \left( \left( t^{(k)} - \sigma\left( 2\mathbf{w} \cdot \mathbf{x}^{(k)} \right) \right) \left( \sigma\left( 2\mathbf{w} \cdot \mathbf{x}^{(k)} \right) \left( 1 - \sigma\left( 2\mathbf{w} \cdot \mathbf{x}^{(k)} \right) \right) \right) \mathbf{x}^{(k)} \right) \\
&= \mathbf{w} + 2\eta \left( \left( t^{(1)} - \sigma\left( 2\mathbf{w} \cdot \mathbf{x}^{(1)} \right) \right) \left( \sigma\left( 2\mathbf{w} \cdot \mathbf{x}^{(1)} \right) \left( 1 - \sigma\left( 2\mathbf{w} \cdot \mathbf{x}^{(1)} \right) \right) \right) \mathbf{x}^{(1)} \right) + \\
&+ 2\eta \left( \left( t^{(2)} - \sigma\left( 2\mathbf{w} \cdot \mathbf{x}^{(2)} \right) \right) \left( \sigma\left( 2\mathbf{w} \cdot \mathbf{x}^{(2)} \right) \left( 1 - \sigma\left( 2\mathbf{w} \cdot \mathbf{x}^{(2)} \right) \right) \right) \mathbf{x}^{(2)} \right) + \\
&+ 2\eta \left( \left( t^{(3)} - \sigma\left( 2\mathbf{w} \cdot \mathbf{x}^{(3)} \right) \right) \left( \sigma\left( 2\mathbf{w} \cdot \mathbf{x}^{(3)} \right) \left( 1 - \sigma\left( 2\mathbf{w} \cdot \mathbf{x}^{(3)} \right) \right) \right) \mathbf{x}^{(3)} \right) + \\
&+ 2\eta \left( \left( t^{(4)} - \sigma\left( 2\mathbf{w} \cdot \mathbf{x}^{(4)} \right) \right) \left( \sigma\left( 2\mathbf{w} \cdot \mathbf{x}^{(4)} \right) \left( 1 - \sigma\left( 2\mathbf{w} \cdot \mathbf{x}^{(4)} \right) \right) \right) \mathbf{x}^{(4)} \right) = \\
&= \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} + \\
&+ 2 \left( 1 - \sigma \left( 2 \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \right) \right) \sigma \left( 2 \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \right) \left( 1 - \sigma \left( 2 \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \right) \right) \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \\
&+ 2 \left( 1 - \sigma \left( 2 \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix} \right) \right) \sigma \left( 2 \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix} \right) \left( 1 - \sigma \left( 2 \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix} \right) \right) \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix} + \\
&+ 2 \left( 0 - \sigma \left( 2 \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \\ 3 \end{pmatrix} \right) \right) \sigma \left( 2 \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \\ 3 \end{pmatrix} \right) \left( 1 - \sigma \left( 2 \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \\ 3 \end{pmatrix} \right) \right) \begin{pmatrix} 1 \\ 1 \\ 3 \end{pmatrix} + \\
&+ 2 \left( 0 - \sigma \left( 2 \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 3 \\ 3 \end{pmatrix} \right) \right) \sigma \left( 2 \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 3 \\ 3 \end{pmatrix} \right) \left( 1 - \sigma \left( 2 \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 3 \\ 3 \end{pmatrix} \right) \right) \begin{pmatrix} 1 \\ 3 \\ 3 \end{pmatrix} = \\
&= \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} + 2 \left( 1 - \sigma \left( 6 \right) \right) \left( \sigma \left( 6 \right) \left( 1 - \sigma \left( 6 \right) \right) \right) \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} + 2 \left( 1 - \sigma \left( 8 \right) \right) \left( \sigma \left( 8 \right) \left( 1 - \sigma \left( 8 \right) \right) \right) \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix} + \\
&+ 2 \left( 0 - \sigma \left( 10 \right) \right) \left( \sigma \left( 10 \right) \left( 1 - \sigma \left( 10 \right) \right) \right) \begin{pmatrix} 1 \\ 1 \\ 3 \end{pmatrix} + 2 \left( 0 - \sigma \left( 14 \right) \right) \left( \sigma \left( 14 \right) \left( 1 - \sigma \left( 14 \right) \right) \right) \begin{pmatrix} 1 \\ 3 \\ 3 \end{pmatrix} = \\
&= \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} + \begin{pmatrix} 1.2197 \times 10^{-5} \\ 1.2197 \times 10^{-5} \\ 1.2197 \times 10^{-5} \end{pmatrix} + \begin{pmatrix} 2.2484 \times 10^{-7} \\ 4.4969 \times 10^{-7} \\ 2.2484 \times 10^{-7} \end{pmatrix} + \\
&+ \begin{pmatrix} -9.0787 \times 10^{-5} \\ -9.0787 \times 10^{-5} \\ -2.7236 \times 10^{-4} \end{pmatrix} + \begin{pmatrix} -1.6631 \times 10^{-6} \\ -4.9892 \times 10^{-6} \\ -4.9892 \times 10^{-6} \end{pmatrix} = \\
&= \begin{pmatrix} 0.99991997 \\ 0.99991687 \\ 0.99973507 \end{pmatrix}
\end{aligned}
$$

---

c) Compute the first stochastic gradient descent update assuming an initialization of all ones.

---

**Solution:**

In stochastic gradient descent we make one update for each training example. So, instead of summing accross all data points we adapt the learning rule for one example only:

$$\mathbf{w} = \mathbf{w} + 2\eta\left(\left(t - \sigma\left(2\mathbf{w}\cdot\mathbf{x}\right)\right)\left(\sigma\left(2\mathbf{w}\cdot\mathbf{x}\right)\left(1-\sigma\left(2\mathbf{w}\cdot\mathbf{x}\right)\right)\right)\mathbf{x}\right)$$

We can now do the updates. Let us start with the first example:

$$\begin{aligned}
&\mathbf{w} = \mathbf{w} + 2\eta\left(\left(t - \sigma\left(2\mathbf{w}\cdot\mathbf{x}\right)\right)\left(\sigma\left(2\mathbf{w}\cdot\mathbf{x}\right)\left(1-\sigma\left(2\mathbf{w}\cdot\mathbf{x}\right)\right)\right)\mathbf{x}\right)\\
&= \begin{pmatrix}1\\1\\1\end{pmatrix} +\\
&+ 2\left(1-\sigma\left(2\begin{pmatrix}1\\1\\1\end{pmatrix}\cdot\begin{pmatrix}1\\1\\1\end{pmatrix}\right)\right)\sigma\left(2\begin{pmatrix}1\\1\\1\end{pmatrix}\cdot\begin{pmatrix}1\\1\\1\end{pmatrix}\right)\left(1-\sigma\left(2\begin{pmatrix}1\\1\\1\end{pmatrix}\cdot\begin{pmatrix}1\\1\\1\end{pmatrix}\right)\right)\begin{pmatrix}1\\1\\1\end{pmatrix}\\
&= \begin{pmatrix}1\\1\\1\end{pmatrix} + 2\left(1-\sigma\left(6\right)\right)\left(\sigma\left(6\right)\left(1-\sigma\left(6\right)\right)\right)\begin{pmatrix}1\\1\\1\end{pmatrix}\\
&= \begin{pmatrix}1\\1\\1\end{pmatrix} + \begin{pmatrix}1.2197\times10^{-5}\\1.2197\times10^{-5}\\1.2197\times10^{-5}\end{pmatrix}\\
&= \begin{pmatrix}1.0000122\\1.0000122\\1.0000122\end{pmatrix}
\end{aligned}$$

---

**2)** Consider the following training data:

$$\left\{\mathbf{x}^{(1)} = \begin{pmatrix}1\\1\end{pmatrix}, \mathbf{x}^{(2)} = \begin{pmatrix}2\\1\end{pmatrix}, \mathbf{x}^{(3)} = \begin{pmatrix}1\\3\end{pmatrix}, \mathbf{x}^{(4)} = \begin{pmatrix}3\\3\end{pmatrix}\right\}$$

$$\left\{t^{(1)} = 1, t^{(2)} = 1, t^{(3)} = 0, t^{(4)} = 0\right\}$$

In this exercise, we will work with a unit that computes the following function:

$$output\left(\mathbf{x};\mathbf{w}\right) = \frac{1}{1+\exp\left(-\mathbf{w}\cdot\mathbf{x}\right)}$$

And we will use the cross-entropy loss function:

$$E(\mathbf{w}) = -\log(p(\mathbf{t} \mid \mathbf{w})) = -\sum_{k=1}^{N} \left( t^{(k)} \log output^{(k)} \left( \mathbf{x}^{(k)}; \mathbf{w} \right) + \left( 1 - t^{(k)} \right) \log \left( 1 - output^{(k)} \left( \mathbf{x}^{(k)}; \mathbf{w} \right) \right) \right)$$

a) Determine the gradient descent learning rule for this unit.

---

**Solution:**

To apply gradient descent, we want an update rule that moves a step of size $\eta$ towards the opposite direction from the gradient of the error function with respect to the weights:

$$\mathbf{w} = \mathbf{w} - \eta \frac{\partial E(\mathbf{w})}{\partial \mathbf{w}}$$

To find the learning rule, we must compute the gradient. Before doing so, we should notice that the model is computing a sigmoid function, so:

$$output(\mathbf{x}; \mathbf{w}) = \frac{1}{1 + \exp(-\mathbf{w} \cdot \mathbf{x})} = \sigma(\mathbf{w} \cdot \mathbf{x})$$

Which has a well-known derivative that we computed earlier:

$$\frac{\partial \sigma(x)}{\partial x} = \sigma(x)(1 - \sigma(x))$$

Having made this auxiliar computation, it is now easier to compute the derivative of the error function with respect to the parameter vector.

$$
\begin{aligned}
\frac{\partial E\left(\mathbf{w}\right)}{\partial \mathbf{w}} &= \frac{\partial\left(-\sum_{k=1}^{N}\left(t^{(k)}\log output^{(k)}\left(\mathbf{x}^{(k)};\mathbf{w}\right)+\left(1-t^{(k)}\right)\log\left(1-output^{(k)}\left(\mathbf{x}^{(k)};\mathbf{w}\right)\right)\right)\right)}{\partial \mathbf{w}} \\
&= -\sum_{k=1}^{N}\frac{\partial\left(t^{(k)}\log output^{(k)}\left(\mathbf{x}^{(k)};\mathbf{w}\right)+\left(1-t^{(k)}\right)\log\left(1-output^{(k)}\left(\mathbf{x}^{(k)};\mathbf{w}\right)\right)\right)}{\partial \mathbf{w}} \\
&= -\sum_{k=1}^{N}\frac{\partial\left(t^{(k)}\log\sigma\left(\mathbf{w}\cdot\mathbf{x}^{(k)}\right)+\left(1-t^{(k)}\right)\log\left(1-\sigma\left(\mathbf{w}\cdot\mathbf{x}^{(k)}\right)\right)\right)}{\partial \mathbf{w}} \\
&= -\sum_{k=1}^{N}\left(\frac{\partial\left(t^{(k)}\log\sigma\left(\mathbf{w}\cdot\mathbf{x}^{(k)}\right)\right)}{\partial \mathbf{w}}+\frac{\partial\left(\left(1-t^{(k)}\right)\log\left(1-\sigma\left(\mathbf{w}\cdot\mathbf{x}^{(k)}\right)\right)\right)}{\partial \mathbf{w}}\right) \\
&= -\sum_{k=1}^{N}\left(t^{(k)}\frac{\partial\left(\log\sigma\left(\mathbf{w}\cdot\mathbf{x}^{(k)}\right)\right)}{\partial \mathbf{w}}+\left(1-t^{(k)}\right)\frac{\partial\left(\log\left(1-\sigma\left(\mathbf{w}\cdot\mathbf{x}^{(k)}\right)\right)\right)}{\partial \mathbf{w}}\right) \\
&= -\sum_{k=1}^{N}(t^{(k)}\frac{\partial\left(\log\sigma\left(\mathbf{w}\cdot\mathbf{x}^{(k)}\right)\right)}{\partial\sigma\left(\mathbf{w}\cdot\mathbf{x}^{(k)}\right)}\frac{\partial\sigma\left(\mathbf{w}\cdot\mathbf{x}^{(k)}\right)}{\partial\left(\mathbf{w}\cdot\mathbf{x}^{(k)}\right)}\frac{\partial\mathbf{w}\cdot\mathbf{x}^{(k)}}{\partial \mathbf{w}}+ \\
&+\left(1-t^{(k)}\right)\frac{\partial\left(\log\left(1-\sigma\left(\mathbf{w}\cdot\mathbf{x}^{(k)}\right)\right)\right)}{\partial\left(1-\sigma\left(\mathbf{w}\cdot\mathbf{x}^{(k)}\right)\right)}\frac{\partial\left(1-\sigma\left(\mathbf{w}\cdot\mathbf{x}^{(k)}\right)\right)}{\partial\sigma\left(\mathbf{w}\cdot\mathbf{x}^{(k)}\right)}\frac{\partial\sigma\left(\mathbf{w}\cdot\mathbf{x}^{(k)}\right)}{\partial\left(\mathbf{w}\cdot\mathbf{x}^{(k)}\right)}\frac{\partial\left(\mathbf{w}\cdot\mathbf{x}^{(k)}\right)}{\partial \mathbf{w}}) \\
&= -\sum_{k=1}^{N}(t^{(k)}\frac{1}{\sigma\left(\mathbf{w}\cdot\mathbf{x}^{(k)}\right)}\sigma\left(\mathbf{w}\cdot\mathbf{x}^{(k)}\right)\left(1-\sigma\left(\mathbf{w}\cdot\mathbf{x}^{(k)}\right)\right)\mathbf{x}^{(k)}+ \\
&+\left(1-t^{(k)}\right)\frac{1}{1-\sigma\left(\mathbf{w}\cdot\mathbf{x}^{(k)}\right)}\left(-1\right)\sigma\left(\mathbf{w}\cdot\mathbf{x}^{(k)}\right)\left(1-\sigma\left(\mathbf{w}\cdot\mathbf{x}^{(k)}\right)\right)\mathbf{x}^{(k)}) \\
&= -\sum_{k=1}^{N}\left(t^{(k)}\left(1-\sigma\left(\mathbf{w}\cdot\mathbf{x}^{(k)}\right)\right)\mathbf{x}^{(k)}+\left(1-t^{(k)}\right)\left(-1\right)\sigma\left(\mathbf{w}\cdot\mathbf{x}^{(k)}\right)\mathbf{x}^{(k)}\right) \\
&= -\sum_{k=1}^{N}\left(t^{(k)}\left(1-\sigma\left(\mathbf{w}\cdot\mathbf{x}^{(k)}\right)\right)\mathbf{x}^{(k)}-\left(1-t^{(k)}\right)\sigma\left(\mathbf{w}\cdot\mathbf{x}^{(k)}\right)\mathbf{x}^{(k)}\right) \\
&= -\sum_{k=1}^{N}\mathbf{x}^{(k)}\left(t^{(k)}\left(1-\sigma\left(\mathbf{w}\cdot\mathbf{x}^{(k)}\right)\right)-\left(1-t^{(k)}\right)\sigma\left(\mathbf{w}\cdot\mathbf{x}^{(k)}\right)\right) \\
&= -\sum_{k=1}^{N}\mathbf{x}^{(k)}\left(t^{(k)}-t^{(k)}\sigma\left(\mathbf{w}\cdot\mathbf{x}^{(k)}\right)-\left(\sigma\left(\mathbf{w}\cdot\mathbf{x}^{(k)}\right)-t^{(k)}\sigma\left(\mathbf{w}\cdot\mathbf{x}^{(k)}\right)\right)\right) \\
&= -\sum_{k=1}^{N}\mathbf{x}^{(k)}\left(t^{(k)}-t^{(k)}\sigma\left(\mathbf{w}\cdot\mathbf{x}^{(k)}\right)-\sigma\left(\mathbf{w}\cdot\mathbf{x}^{(k)}\right)+t^{(k)}\sigma\left(\mathbf{w}\cdot\mathbf{x}^{(k)}\right)\right) \\
&= -\sum_{k=1}^{N}\mathbf{x}^{(k)}\left(t^{(k)}-\sigma\left(\mathbf{w}\cdot\mathbf{x}^{(k)}\right)\right)
\end{aligned}
$$

So, we can write our update rule as follows.

$$
\begin{aligned}
\mathbf{w} &= \mathbf{w} - \eta \frac{\partial E\left(\mathbf{w}\right)}{\partial \mathbf{w}} \\
&= \mathbf{w} - \eta \left( - \sum_{k=1}^{N} \mathbf{x}^{(k)} \left( t^{(k)} - \sigma \left( \mathbf{w} \cdot \mathbf{x}^{(k)} \right) \right) \right) \\
&= \mathbf{w} + \eta \sum_{k=1}^{N} \mathbf{x}^{(k)} \left( t^{(k)} - \sigma \left( \mathbf{w} \cdot \mathbf{x}^{(k)} \right) \right)
\end{aligned}
$$

---

b) Compute the first gradient descent update assuming an initialization of all ones .

---

**Solution:**

The original gradient descent does one update per epoch because its learning rule requires contributions from all data points to do one step. Let us compute it.

According to the problem statement, we start with weights $\mathbf{w} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$ and learning rate $\eta = 1.0$.

$$
\begin{aligned}
\mathbf{w} &= \mathbf{w} + \eta \sum_{k=1}^{N} \mathbf{x}^{(k)} \left( t^{(k)} - \sigma \left( \mathbf{w} \cdot \mathbf{x}^{(k)} \right) \right) \\
&= \mathbf{w} + \eta \sum_{k=1}^{4} \mathbf{x}^{(k)} \left( t^{(k)} - \sigma \left( \mathbf{w} \cdot \mathbf{x}^{(k)} \right) \right) \\
&= \mathbf{w} + \eta \mathbf{x}^{(1)} \left( t^{(1)} - \sigma \left( \mathbf{w} \cdot \mathbf{x}^{(1)} \right) \right) + \eta \mathbf{x}^{(2)} \left( t^{(2)} - \sigma \left( \mathbf{w} \cdot \mathbf{x}^{(2)} \right) \right) + \\
&+ \eta \mathbf{x}^{(3)} \left( t^{(3)} - \sigma \left( \mathbf{w} \cdot \mathbf{x}^{(3)} \right) \right) + \eta \mathbf{x}^{(4)} \left( t^{(4)} - \sigma \left( \mathbf{w} \cdot \mathbf{x}^{(4)} \right) \right) \\
&= \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \left( 1 - \sigma \left( \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \right) \right) + \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix} \left( 1 - \sigma \left( \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix} \right) \right) + \\
&+ \begin{pmatrix} 1 \\ 1 \\ 3 \end{pmatrix} \left( 0 - \sigma \left( \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \\ 3 \end{pmatrix} \right) \right) + \begin{pmatrix} 1 \\ 3 \\ 3 \end{pmatrix} \left( 0 - \sigma \left( \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 3 \\ 3 \end{pmatrix} \right) \right) \\
&= \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} (1 - \sigma(3)) + \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix} (1 - \sigma(4)) + \begin{pmatrix} 1 \\ 1 \\ 3 \end{pmatrix} (0 - \sigma(5)) + \begin{pmatrix} 1 \\ 3 \\ 3 \end{pmatrix} (0 - \sigma(7)) \\
&= \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} + \begin{pmatrix} 0.0474 \\ 0.0474 \\ 0.0474 \end{pmatrix} + \begin{pmatrix} 0.0179 \\ 0.0359 \\ 0.0179 \end{pmatrix} + \begin{pmatrix} -0.9933 \\ -0.9933 \\ -2.9799 \end{pmatrix} + \begin{pmatrix} -0.9991 \\ -2.9973 \\ -2.9973 \end{pmatrix} \\
&= \begin{pmatrix} -0.9269 \\ -2.9072 \\ -4.9118 \end{pmatrix}
\end{aligned}
$$

---

c) Compute the first stochastic gradient descent update assuming an initialization of all ones.

---

**Solution:**

In stochastic gradient descent we make one update for each training example. So, instead of summing accross all data points we adapt the learning rule for one example only:

$$\mathbf{w} = \mathbf{w} + \eta \mathbf{x} \left( t - \sigma \left( \mathbf{w} \cdot \mathbf{x} \right) \right)$$

We can now do the updates. Let us start with the first example:

$$
\begin{aligned}
\mathbf{w} &= \mathbf{w} + \eta \mathbf{x} \left(t - \sigma \left(\mathbf{w} \cdot \mathbf{x}\right)\right) \\
&= \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} + 1 \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \left(1 - \sigma \left(\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}\right)\right) \\
&= \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \left(1 - \sigma \left(3\right)\right) \\
&= \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} + \begin{pmatrix} 0.0474 \\ 0.0474 \\ 0.0474 \end{pmatrix} \\
&= \begin{pmatrix} 1.0474 \\ 1.0474 \\ 1.0474 \end{pmatrix}
\end{aligned}
$$

---

**3)** Consider the following training data:

$$\left\{\mathbf{x}^{(1)} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \mathbf{x}^{(2)} = \begin{pmatrix} 2 \\ 1 \end{pmatrix}, \mathbf{x}^{(3)} = \begin{pmatrix} 1 \\ 3 \end{pmatrix}, \mathbf{x}^{(4)} = \begin{pmatrix} 3 \\ 3 \end{pmatrix}\right\}$$

$$\left\{t^{(1)} = 1, t^{(2)} = 1, t^{(3)} = 0, t^{(4)} = 0\right\}$$

In this exercise, we will work with a unit that computes the following function:

$$\mathit{output}\left(\mathbf{x}; \mathbf{w}\right) = \exp\left(\left(\mathbf{w} \cdot \mathbf{x}\right)^2\right)$$

And we will use the half sum of squared errors as our error (loss) function:

$$E\left(\mathbf{w}\right) = \frac{1}{2} \sum_{k=1}^{N} \left(t^{(k)} - \mathit{output}\left(\mathbf{x}^{(k)}; \mathbf{w}\right)\right)$$

a) Determine the gradient descent learning rule for this unit.

---

**Solution:**

To apply gradient descent, we want an update rule that moves a step of size $\eta$ towards the opposite direction from the gradient of the error function with respect to the weights:

$$\mathbf{w} = \mathbf{w} - \eta \frac{\partial E\left(\mathbf{w}\right)}{\partial \mathbf{w}}$$

To find the learning rule, we must compute the gradient of the error function with respect to the parameter vector.

$$\begin{aligned}
\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}} &= \frac{\partial \frac{1}{2}\sum_{k=1}^{N}\left(t^{(k)} - output\left(\mathbf{x}^{(k)};\mathbf{w}\right)\right)^2}{\partial \mathbf{w}} \\
&= \frac{\partial \frac{1}{2}\sum_{k=1}^{N}\left(t^{(k)} - \exp\left(\left(\mathbf{w}\cdot\mathbf{x}^{(k)}\right)^2\right)\right)^2}{\partial \mathbf{w}} \\
&= \frac{1}{2}\frac{\partial \sum_{k=1}^{N}\left(t^{(k)} - \exp\left(\left(\mathbf{w}\cdot\mathbf{x}^{(k)}\right)^2\right)\right)^2}{\partial \mathbf{w}} \\
&= \frac{1}{2}\sum_{k=1}^{N}\frac{\partial\left(t^{(k)} - \exp\left(\left(\mathbf{w}\cdot\mathbf{x}^{(k)}\right)^2\right)\right)^2}{\partial \mathbf{w}} \\
&= \frac{1}{2}\sum_{k=1}^{N}\frac{\partial\left(t^{(k)} - \exp\left(\left(\mathbf{w}\cdot\mathbf{x}^{(k)}\right)^2\right)\right)^2}{\partial\left(t^{(k)} - \exp\left(\left(\mathbf{w}\cdot\mathbf{x}^{(k)}\right)^2\right)\right)}\frac{\partial\left(t^{(k)} - \exp\left(\left(\mathbf{w}\cdot\mathbf{x}^{(k)}\right)^2\right)\right)}{\partial \exp\left(\left(\mathbf{w}\cdot\mathbf{x}^{(k)}\right)^2\right)} \\
&\quad \frac{\partial \exp\left(\left(\mathbf{w}\cdot\mathbf{x}^{(k)}\right)^2\right)}{\partial\left(\left(\mathbf{w}\cdot\mathbf{x}^{(k)}\right)^2\right)}\frac{\partial\left(\left(\mathbf{w}\cdot\mathbf{x}^{(k)}\right)^2\right)}{\partial\left(\mathbf{w}\cdot\mathbf{x}^{(k)}\right)}\frac{\partial\left(\mathbf{w}\cdot\mathbf{x}^{(k)}\right)}{\partial \mathbf{w}} \\
&= \frac{1}{2}\sum_{k=1}^{N}\left(2\left(t^{(k)} - \exp\left(\left(\mathbf{w}\cdot\mathbf{x}^{(k)}\right)^2\right)\right)(-1)\exp\left(\left(\mathbf{w}\cdot\mathbf{x}^{(k)}\right)^2\right)2\left(\mathbf{w}\cdot\mathbf{x}^{(k)}\right)\mathbf{x}^{(k)}\right) \\
&= -2\sum_{k=1}^{N}\left(\left(t^{(k)} - \exp\left(\left(\mathbf{w}\cdot\mathbf{x}^{(k)}\right)^2\right)\right)\exp\left(\left(\mathbf{w}\cdot\mathbf{x}^{(k)}\right)^2\right)\left(\mathbf{w}\cdot\mathbf{x}^{(k)}\right)\mathbf{x}^{(k)}\right)
\end{aligned}$$

So, we can write our update rule as follows.

$$\begin{aligned}
\mathbf{w} &= \mathbf{w} - \eta\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}} \\
&= \mathbf{w} - \eta\left(-2\sum_{k=1}^{N}\left(\left(t^{(k)} - \exp\left(\left(\mathbf{w}\cdot\mathbf{x}^{(k)}\right)^2\right)\right)\exp\left(\left(\mathbf{w}\cdot\mathbf{x}^{(k)}\right)^2\right)\left(\mathbf{w}\cdot\mathbf{x}^{(k)}\right)\mathbf{x}^{(k)}\right)\right) \\
&= \mathbf{w} + 2\eta\sum_{k=1}^{N}\left(\left(t^{(k)} - \exp\left(\left(\mathbf{w}\cdot\mathbf{x}^{(k)}\right)^2\right)\right)\exp\left(\left(\mathbf{w}\cdot\mathbf{x}^{(k)}\right)^2\right)\left(\mathbf{w}\cdot\mathbf{x}^{(k)}\right)\mathbf{x}^{(k)}\right)
\end{aligned}$$

---

b) Compute the stochastic gradient descent update for input $\mathbf{x} = \begin{pmatrix}1\\1\end{pmatrix}, t = 0$ initialized with $\mathbf{w} = \begin{pmatrix}0\\1\\0\end{pmatrix}$ and learning rate $\eta = 2$.

---

**Solution:**

In stochastic gradient descent we make one update for each training example. So, instead of summing accross all data points we adapt the learning rule for one example only:

$$\mathbf{w} = \mathbf{w} + 2\eta \left( \left( t - \exp\left( (\mathbf{w} \cdot \mathbf{x})^2 \right) \right) \exp\left( (\mathbf{w} \cdot \mathbf{x})^2 \right) (\mathbf{w} \cdot \mathbf{x}) \, \mathbf{x} \right)$$

We can now do the updates. Let us start with the first example:

$$\begin{aligned}
&\mathbf{w} = \mathbf{w} + 2\eta \left( \left( t - \exp\left( (\mathbf{w} \cdot \mathbf{x})^2 \right) \right) \exp\left( (\mathbf{w} \cdot \mathbf{x})^2 \right) (\mathbf{w} \cdot \mathbf{x}) \, \mathbf{x} \right) \\
&= \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} + 2\,(2) \left( 0 - \exp\left( \left( \left( \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \right)^2 \right) \right) \right) \\
&\exp\left( \left( \left( \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \right)^2 \right) \left( \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \right) \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \right) \\
&= \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} + 4 \left( (0 - \exp(1)) \exp(1)\,(1) \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \right) \\
&= \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} - 4 \begin{pmatrix} e^2 \\ e^2 \\ e^2 \end{pmatrix} \\
&= \begin{pmatrix} -4e^2 \\ 1 - 4e^2 \\ -4e^2 \end{pmatrix}
\end{aligned}$$

---

## 3 Thinking Questions

a) Until now we could only solve classification tasks where the two classes were separated by simple lines. Now we have seen that we can apply any feature transformations we want. Think about which kinds of problems we can solve now? Is it all a matter of finding the right transformation? Is it easy to choose the right transformation?

b) When using the linear regression for classification, think about how the threshold changes the sensitivity of the model. Is it more or less likely that the model will fail to recognize a class member as the threshold increases?

c) Think about the error functions we have seen. Do you think that one is clearly better than the other? What changes when one changes the error function?