Tutorial Business Analytics

Exercise 12

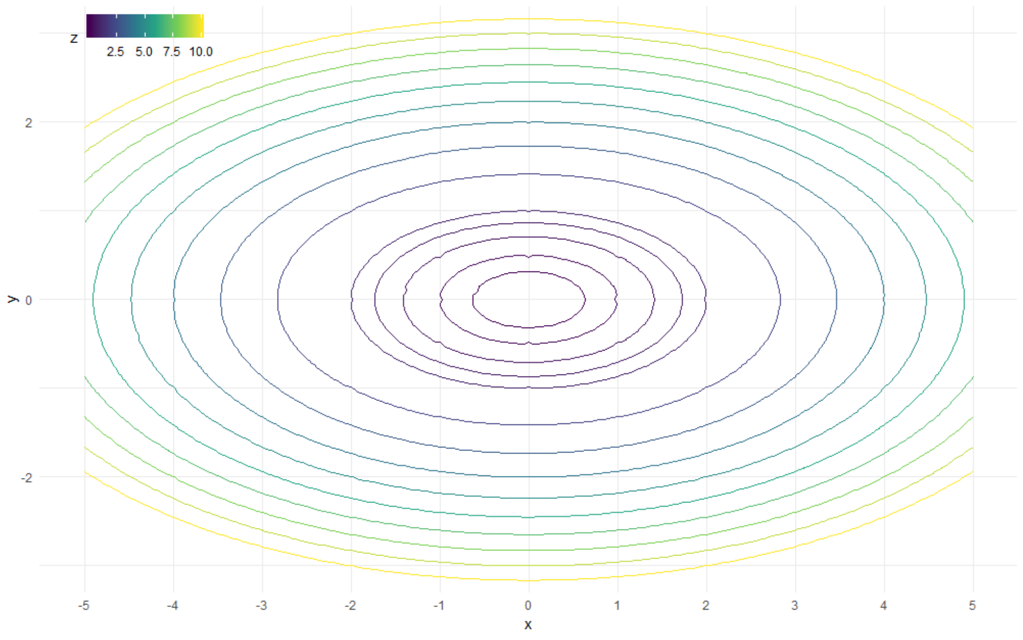
Exercise 12.1 Gradient Descent

Let $f: \mathbb{R}^2 \rightarrow \mathbb{R}$ be the convex function given by

$$f(x,y) = \frac{x^2}{4} + y^2$$

a) Find $\nabla f(x,y)$.

b) Starting from $(x_0, y_0) = (3,2)^T$ perform 3 steps of gradient descent with a learning rate of $\alpha = 1$. Plot your gradient steps. What do you observe? Does the function value decrease in each step? Will this sequence converge to the optimum at $(0,0)$?



c) Repeat c) but with a learning rate rule that is guaranteed to converge:

$$\alpha_n = \frac{1}{n}$$

d) Starting from (x_1, y_1) found in b), perform 2 additional steps of the *momentum method*, with $\beta = 0.25$ and $\alpha = 1$. Assume that $d_1 = \nabla f(x_0, y_0)$

Recall from tutorial slides:

$$d_n = \beta d_{n-1} + \alpha \nabla f(x_{n-1}), \qquad x_n = x_{n-1} - d_n.$$

Exercise 12.2 Backpropagation I

Consider the following feed-forward neural network that consists of

- An input layer ($l = 0$) representing two-dimensional points

$$a^{[0]} = \left(a_1^{[0]}, a_2^{[0]}\right)^T \in \mathbb{R}^2$$

- A hidden layer $l = 1$ with 2 hidden nodes and sigmoid activation function $g^{[1]}$
- An output layer $l = 2$ with one node and sigmoid activation function $g^{[2]}$.

a) Write down the formulas for the forward pass of this neural network.
How many trainable parameters does it have?

b) In the following, we will train the NN for binary classification on a data set. For a given input-output pair (x, y), we will use x as the input to the NN, model $y \approx \hat{y} = a^{[2]}(x)$ and evaluate the model using the *cross-entropy loss*

$$\ell(y, \hat{y}) = -[y \ln \hat{y} + (1 - y) \ln(1 - \hat{y})]$$

Calculate the partial derivatives $\frac{\partial \ell}{\partial W_1^{[2]}}$ and $\frac{\partial \ell}{\partial W_2^{[2]}}$ that will be used to update $W^{[2]}$ in backpropagation.

Hint: You may use the following derivative of the sigmoid function $\sigma(\cdot)$ without proof: $\sigma'(x) = \sigma(x) \cdot (1 - \sigma(x))$

Tutorial Business Analytics

Exercise 12 - Solution

Exercise 12.1 Gradient Descent

Let $f\colon \mathbb{R}^2 \to \mathbb{R}$ be the convex function given by

$$f(x,y) = \frac{x^2}{4} + y^2$$

a) Find $\nabla f(x,y)$.

$$\nabla f(x,y) = \begin{pmatrix} \partial f(x,y)/\partial x \\ \partial f(x,y)/\partial y \end{pmatrix} = \begin{pmatrix} 0.5\, x \\ 2\, y \end{pmatrix}$$

b) Starting from $(x_0, y_0) = (3,2)^T$ perform 3 steps of gradient descent with a learning rate of $\alpha = 1$. Plot your gradient steps. What do you observe? Does the function value decrease in each step? Will this sequence converge to the optimum at (0,0)?

For later reference, our original function value is $f(x_0, y_0) = \frac{3^2}{4} + 2^2 = 6.25$

In the case of two-dimensional input vectors $(x,y)^T$, the gradient update rule is $\begin{pmatrix} x_n \\ y_n \end{pmatrix} = \begin{pmatrix} x_{n-1} \\ y_{n-1} \end{pmatrix} - \alpha \nabla f(x_{n-1}, y_{n-1})$, thus we calculate

$$\nabla f(x_0, y_0) = \begin{pmatrix} 0.5 \cdot 3 \\ 2 \cdot 2 \end{pmatrix} = \begin{pmatrix} 0.5 \cdot 3 \\ 2 \cdot 2 \end{pmatrix} = \begin{pmatrix} 1.5 \\ 4 \end{pmatrix},$$

$$\begin{pmatrix} x_1 \\ y_1 \end{pmatrix} = \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} - \alpha \cdot \nabla f \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} = \begin{pmatrix} 3 \\ 2 \end{pmatrix} - 1 \cdot \begin{pmatrix} 1.5 \\ 4 \end{pmatrix} = \begin{pmatrix} 1.5 \\ -2 \end{pmatrix}.$$

Just for comparison, the new function value is then

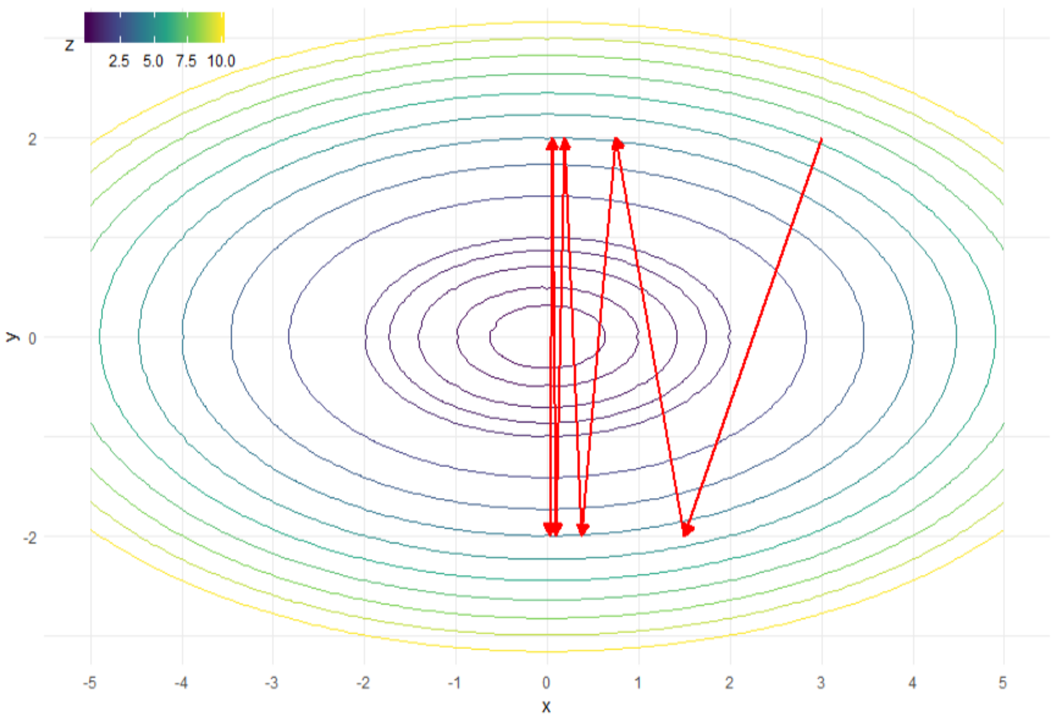
$$f\begin{pmatrix} x_1 \\ y_1 \end{pmatrix} = \frac{1.5^2}{4} + (-2)^2 = 4.5625$$

Repeating these steps, using the update formula above, we get

$$\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = \begin{pmatrix} 1.5 \\ -2 \end{pmatrix} - 1 \cdot \begin{pmatrix} 0.75 \\ -4 \end{pmatrix} = \begin{pmatrix} 0.75 \\ 2 \end{pmatrix}, \qquad f\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} \approx 4.141$$

$$\begin{pmatrix} x_3 \\ y_3 \end{pmatrix} = \begin{pmatrix} 0.75 \\ 2 \end{pmatrix} - 1 \cdot \begin{pmatrix} 0.375 \\ 4 \end{pmatrix} = \begin{pmatrix} 0.375 \\ -2 \end{pmatrix}, \qquad f\begin{pmatrix} x_3 \\ y_3 \end{pmatrix} \approx 4.035$$

However, visualizing the first 7 steps, we see that the sequence does not converge to the minimum at $(0,0)^T$.



In fact,

$$\nabla f(0,2) = \binom{0}{4} = -\nabla f(0,-2),$$

and thus, in the limit, our method will oscillate between these two points, never achieving a function value of less than 4 and never reaching the optimum.

c) Repeat b) but with a learning rate rule that is guaranteed to converge:

$$\alpha_n = \frac{1}{n}$$

The gradient update formula with dynamic step size α_n is

$$\binom{x_n}{y_n} = \binom{x_{n-1}}{y_{n-1}} - \alpha_n \nabla f(x_{n-1}, y_{n-1}),$$

as $\alpha_1 = \frac{1}{1} = 1$, the first step for calculating $(x_1, y_1)^T$ is identical to b). Afterwards, we get

$$\binom{x_2}{y_2} = \binom{1.5}{-2} - 1/2 \cdot \binom{0.75}{-4} = \binom{1.125}{0}, \qquad f\binom{x_2}{y_2} \approx 0.316$$

$$\binom{x_3}{y_3} \approx \binom{1.125}{0} - 1/3 \cdot \binom{0.563}{0} = \binom{0.936}{0}, \qquad f\binom{x_3}{y_3} \approx 0.220$$

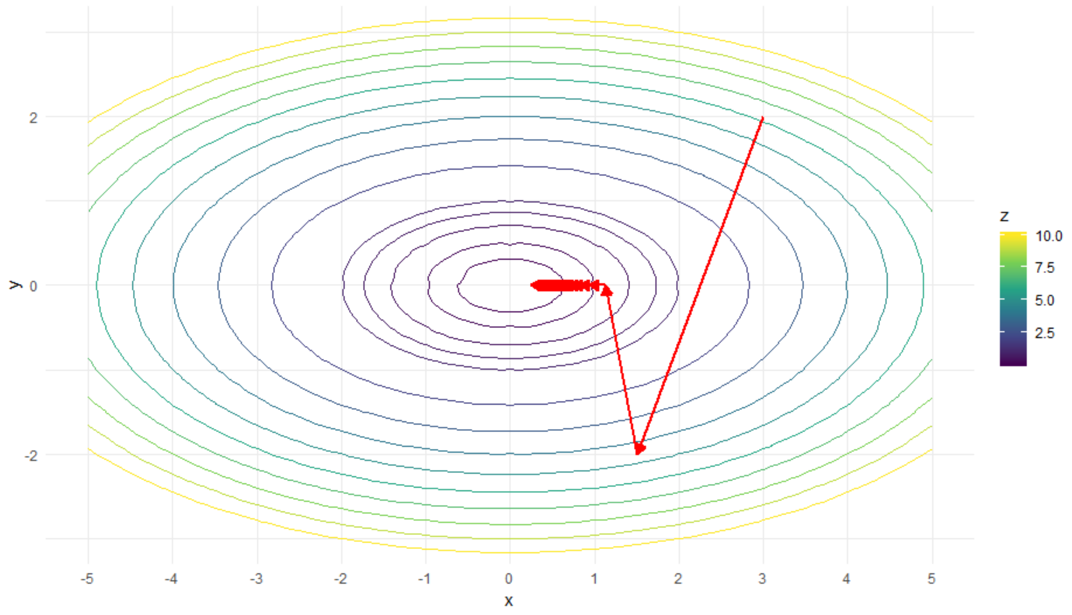
$$\binom{x_4}{y_4} \approx \binom{0.936}{0} - 1/4 \cdot \binom{0.469}{0} = \binom{0.820}{0}, \qquad f\binom{x_4}{y_4} \approx 0.168$$

We see that with this choice, we have broken the oscillation that we saw in the previous exercise. However, we also observe that the first coordinate is diminishing much slower than we saw before with a learning rate of 1.

In fact, if we continue, we will get

$$\begin{pmatrix} x_{50} \\ y_{50} \end{pmatrix} \approx \begin{pmatrix} 0.239 \\ 0 \end{pmatrix}, \quad \begin{pmatrix} x_{100} \\ y_{100} \end{pmatrix} \approx \begin{pmatrix} 0.169 \\ 0 \end{pmatrix}, \quad \begin{pmatrix} x_{100} \\ y_{100} \end{pmatrix} \approx \begin{pmatrix} 0.076 \\ 0 \end{pmatrix}$$

Depending on our requirements, this rate of convergence might be much too slow. Visualization of 50 steps:



d) Starting from (x_1, y_1) in b), perform 2 additional steps of the *momentum method with learning rate decay*, with $\beta = 0.25$ and $\alpha = 1$.

How do the results compare to b) and c)? How does the momentum affect the step-lengths in the x and y components?

Recall from Tutorial slides:

$$d_n = \beta d_{n-1} + \alpha \nabla f(x_{n-1}) \qquad x_n = x_{n-1} - \; d_n.$$

If we had started with $d_0 = (0,0)^T$, calculating (x_1, y_1) is identical to (b) and (c), since

$$d_1 = \beta \cdot 0 + \alpha \nabla f \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} = \nabla f \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} = \begin{pmatrix} 1.5 \\ 4 \end{pmatrix}$$

Continuing, with the second step, we get

$$d_2 = \beta \cdot d_1 + \alpha \, \nabla f \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} = \; 0.25 \cdot \begin{pmatrix} 1.5 \\ 4 \end{pmatrix} + \begin{pmatrix} 0.75 \\ -4 \end{pmatrix} = \begin{pmatrix} 1.125 \\ -3 \end{pmatrix}$$

$$\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} - d_2 = \begin{pmatrix} 1.5 \\ -2 \end{pmatrix} - \begin{pmatrix} 1.125 \\ -3 \end{pmatrix} = \begin{pmatrix} 0.375 \\ 1 \end{pmatrix}, \qquad f \begin{pmatrix} x_2 \\ y_2 \end{pmatrix} \approx 1.035$$

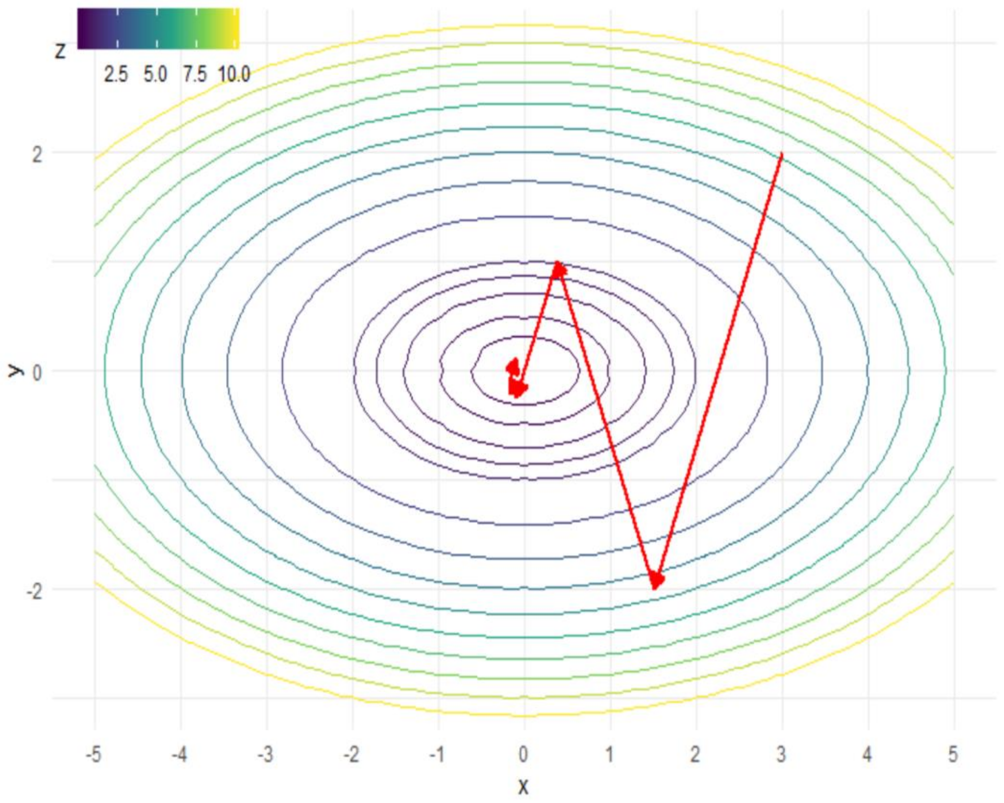
$$d_3 = 0.25 \cdot d_2 + \nabla f \begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = 0.25 \cdot \begin{pmatrix} 1.125 \\ -3 \end{pmatrix} + \begin{pmatrix} 0.1875 \\ 2 \end{pmatrix} \approx \begin{pmatrix} 0.469 \\ 1.25 \end{pmatrix}$$

$$\begin{pmatrix} x_3 \\ y_3 \end{pmatrix} = \begin{pmatrix} x_2 \\ y_2 \end{pmatrix} - d_3 \approx \begin{pmatrix} 0.375 \\ 1 \end{pmatrix} - \begin{pmatrix} 0.469 \\ 1.25 \end{pmatrix} = \begin{pmatrix} -0.094 \\ -0.25 \end{pmatrix}, \qquad f \begin{pmatrix} x_3 \\ y_3 \end{pmatrix} \approx 0.065$$

With momentum, the oscillation from b) has been broken, while also admitting larger step-sizes than those in c). In particular, when d and ∇f point into the same direction, momentum yields to larger step-length than the gradient alone, if they point into opposite directions, momentum shortens the step.

Compared with the steps taken in b), in the second and third step, the momentum term dampened the oscillation in the y component and accelerated the steps in the x component.

5 Steps visualized:



Exercise 12.2 Backpropagation I

Consider the following feed-forward neural network that consists of

- An input layer ($l = 0$) representing two-dimensional points

$$a^{[0]} = \left(a_1^{[0]}, a_2^{[0]}\right)^T \in \mathbb{R}^2$$

- A hidden layer $l = 1$ with 2 hidden nodes and sigmoid activation function $g^{[1]}$
- An output layer $l = 2$ with one node and sigmoid activation function $g^{[2]}$.

a) Write down the formulas for the forward pass of this neural network.
   How many trainable parameters does it have?

We will consider the case of a single input observation. Let $x = \begin{pmatrix} a_1^{[0]} \\ a_2^{[0]} \end{pmatrix}$ be the input vector associated with that observation. Then the forward pass through the neural network can be written using the following four equations:

$$z^{[1]} = W^{[1]}x + b^{[1]}, \qquad a^{[1]} = g^{[1]}\left(z^{[1]}\right) = \sigma\left(z^{[1]}\right)$$

$$z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}, \qquad a^{[2]} = g^{[2]}\left(z^{[2]}\right) = \sigma(z^{[2]})$$

Where $\sigma(z) = \frac{1}{1+\exp(-z)}$ is the sigmoid function. $\hat{y} = a^{[2]}$ is the output of the NN.

The free parameters of the Neural Net are the entries $W^{[1]},\ b^{[1]},\ W^{[2]},\ b^{[2]}$, so the question asks us to find their dimensions.

As the both the input and hidden layer have 2 units, the shapes of both x and $a^{[1]}$ are 2x1 (i.e. column vectors with two entries, the number of rows corresponds to the number of units in the layer, the number of columns (1) corresponds to the number of observations from the data). From this we can deduct that the dimension of $W^{[1]}$ must be 2x2 because the dimensions are fully determined by the fact that the matrix multiplication must have correct dimensions, i.e.

$$\text{first dimension of } W^{[1]} = \text{first dimension of } z^{[1]} = \#\{\text{nodes in layer 1}\} = 2$$
$$\text{second dimension of } W^{[1]} = \text{first dimension } of\ x = \#\{\text{nodes in layer 0}\} = 2$$

Similarly, for the addition to be valid, $b^{[1]}$ must have the same shape as $z^{[1]}$ and $W^{[1]}x$, it its shape is therefore 2x1.

Repeating the same deliberations for the second layer we get shapes of 1x2 for $W^{[2]}$ and 1x1 for $b^{[2]}$.

With parameters

$$W^{[1]} \in \mathbb{R}^{2\times 2},\ b^{[1]} \in \mathbb{R}^2,\ W^{[2]} \in \mathbb{R}^{1\times 2}, \text{ and } b^{[2]} \in \mathbb{R}^1,$$

the network thus has a total of $4+2+2+1=9$ trainable parameters.

b) In the following, we will train the NN for binary classification on a data set. For a given input-output pair (x, y), we will use x as the input to the NN, model $y \approx \hat{y} = a^{[2]}$ and evaluate the model using the *cross-entropy loss*

$$\ell(y, \hat{y}) = -[y \ln \hat{y} + (1-y) \ln(1-\hat{y})]$$

Calculate the partial derivatives $\frac{\partial \ell}{\partial w_1^{[2]}}$ and $\frac{\partial \ell}{\partial w_2^{[2]}}$ that will be used in backpropagation.

Hint: You may use the following derivative of the sigmoid function $\sigma(\cdot)$ without proof: $\sigma'(x) = \sigma(x) \cdot (1 - \sigma(x))$

We apply backpropagation to find the desired derivatives with respect to $W^{[2]}$: While ℓ is not directly expressed as a function of $W^{[2]}$, we have

$$z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}, \quad a^{[2]} = \sigma\left(z^{[2]}\right), \;\; \ell\left(y, a^{[2]}\right) = -[y \ln a + (1-y) \ln(1-a)]$$

We can thus find the desired derivative by applying the chain rule twice:

$$\frac{\partial \ell}{\partial w_1^{[2]}} = \frac{\partial \ell}{\partial a^{[2]}} \cdot \frac{\partial a^{[2]}}{\partial w_1^{[2]}} = \frac{\partial \ell}{\partial a^{[2]}} \cdot \frac{\partial a^{[2]}}{\partial z^{[2]}} \cdot \frac{\partial z^{\;[2]}}{\partial w_1^{[2]}}$$

We calculate these derivatives separately by deriving the three functions above:

$$da^{[2]} = \frac{\partial \ell\left(a^{[2]}, y\right)}{\partial a^{[2]}} = -\left[y \cdot \frac{1}{a^{[2]}} + (1-y)\frac{-1}{1-a^{[2]}}\right] = \frac{1-y}{1-a^{[2]}} - \frac{y}{a^{[2]}}$$

$$\frac{\partial a^{[2]}}{\partial z^{[2]}} = \sigma'\left(z^{[2]}\right) = \sigma\left(z^{[2]}\right) \cdot \left(1 - \sigma\left(z^{[2]}\right)\right) = a^{[2]} \cdot (1 - a^{[2]})$$

where we used the equation given for the sigmoid derivative and the fact that the sigmoid is the activation function of the second layer. We can combine the first two derivatives we calculated to get

$$dz^{[2]} = \frac{\partial \ell}{\partial z^{[2]}} = \frac{\partial \ell}{\partial a^{[2]}} \cdot \frac{\partial a^{[2]}}{\partial z^{[2]}} = \left[\frac{1-y}{1-a^{[2]}} - \frac{y}{a^{[2]}}\right] \cdot a^{[2]} \cdot \left(1 - a^{[2]}\right)$$

$$= a^{[2]}(1-y) - \left(1 - a^{[2]}\right)y = a^{[2]} - a^{[2]}y - y + a^{[2]}y = a^{[2]} - y$$

To calculate the last derivative, we use the fact that $z^{[2]}$ is scalar (only one unit in output layer) to rewrite its equation without matrix notation:

$$z^{[2]} = W^{[2]}a^{[1]} + b^{[2]} = w_1^{[2]}a_1^{[1]} + w_2^{[2]}a_2^{[1]} + b^{[2]}$$

Thus

$$\frac{\partial z^{[2]}}{\partial w_1^{[2]}} = \mathrm{a}_1^{[1]}, \quad \frac{\partial z^{[2]}}{\partial w_2^{[2]}} = \mathrm{a}_2^{[1]}$$

Combining the results of the individual derivatives, we get

$$\begin{aligned} dW^{[2]} &= \begin{pmatrix} \frac{\partial \ell}{\partial w_1^{[2]}} & \frac{\partial \ell}{\partial w_2^{[2]}} \end{pmatrix} = dz^{[2]} \cdot \begin{pmatrix} \frac{\partial z^{[2]}}{\partial w_1^{[2]}} & \frac{\partial z^{[2]}}{\partial w_2^{[2]}} \end{pmatrix} \\ &= \left(a^{[2]} - y\right) \cdot \begin{pmatrix} a_1^{[1]} & a_2^{[1]} \end{pmatrix} = \left(a^{[2]} - y\right) \cdot {a^{[1]}}^T. \end{aligned}$$