CISC 371   Class 14

Neural Networks – Multiple Layers

Article: Rumelhart et al, *Learning internal representations by back-propagating errors* [1]

Main Concepts:

- *A layer is a linear computation within an activation computation*
- *The linear computation is a matrix-vector product*
- *Steepest descent uses Hadamard products*
- *Back-propagation uses scale factors for descent vectors*

Sample Problem, Machine Learning: How can we write a descent method for a multi-layer neural network?

We will extend the methods and results of Class 12 to an arbitrary number p of layers. We will assume that the activation function for each neuron in a layer is the same, i.e., for each neuron in layer l the activation function is ${}_l\phi(\cdot)$.

We will think of a *layer* of artificial neurons as an ordered list of individual neurons. Each neuron in a layer has a common input and a shared activation function $\phi(\cdot)$. Figure 14.1 shows a neural network that has a single hidden layer.

We will use the neural-network convention that the data $\underline{x}$ are provided to Layer 2. This convention implies that we will need to rewrite the mathematical terms of the previous class to have a subscript, e.g., the weight vector of the output neuron is now ${}_3\vec{w}$ in place of $\vec{w}$. In this convention, layers are added to the "output side" of the network.

In this course, we know how to differentiate a vector function of a vector to find a Jacobian matrix. A neural network uses data that are observations, i.e. 1-forms, and weights that are vectors. Because in the intermediate layers, data are dependent on weights of previous layers, we will need to transpose terms in ways that keep data observations distinct from vectors.

We will add, to the single output neuron described in Class 12, a layer of m_2 artificial neurons. These will feed forward to the output neuron, so the number of inputs to the output neuron will be $n_3 = m_2$. The input to Layer 2 will be a data observation ${}_2\underline{x}$ that has n_2 entries, so ${}_2\underline{x} \in \mathbb{R}^{n_2}$. A consequence of adding a layer is that the number of entries in the weight vector for Layer 3 is $n_3 = m_2$.

For Layer 2, we will have to use a bias scalar to find the linear term for the neuron; to do this, we will augment the data parameter with the value 1. This implies that the data 1-form that is the input parameter to this layer of neuron is $[{}_2\underline{x}\ 1]$.

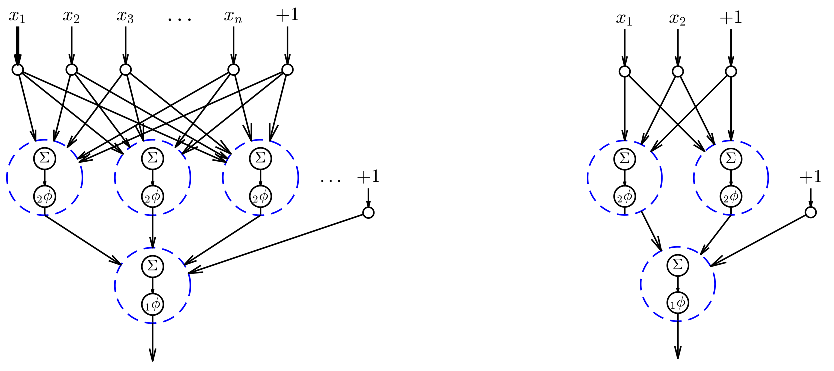

Figure 14.1: A neural net with a single hidden layer. The input layer, Layer 1, has no computation. The output vector of the hidden layer, Layer 2, is augmented as $_3\underline{x}$ and provided to the output neuron, which is Layer 3. The input vector $\vec{x}$ is augmented as $[\vec{x}^T \;\; 1]$ and provided to Layer 2. The outputs of the Layer 2 are augmented and sent to layer 3. (A) The general structure of such a network. (B) A simplified network with size-2 inputs and two neurons in the hidden layer, Layer 2.

The linear term for Layer 2 has m_2 entries, one for each neuron in Layer 2; this is in accordance with Layer 3, where the linear term has one entry because Layer 3 has one neuron. We will write the linear term for Layer 2 $\underline{u} \in \mathbb{R}^{m_2}$. This term is a function of the weights for the layer and has a data parameter that is the input to the layer.

14.1 ANN – Layer 2

Each neuron in Layer 2 has a weight vector with $n_2 + 1$ entries, one each for the entries of the input $_2\underline{x}$ and one for the bias scalar. For the j^{th} neuron, we will write the weight vector as

$$\begin{aligned}
{}_2\vec{w}_j &\in \mathbb{R}^{n_2+1} \\
{}_2\vec{w}_{j,1..n_3} &\overset{\text{def}}{=} \begin{bmatrix} {}_2w_{j,1} \\ {}_2w_{j,2} \\ \vdots \\ {}_2w_{j,n_2} \end{bmatrix} \\
{}_2\vec{w}_j &= \begin{bmatrix} {}_2\vec{w}_{j,1..n_2} \\ {}_2w_{j,n_2+1} \end{bmatrix} \qquad (14.1)
\end{aligned}$$

We will gather these weight vectors in two ways: as a single weight vector for Layer 2, which is ${}_2\vec{w}$, and as a weight matrix for Layer 2, which is ${}_2W$. The weights Layer 2 are

$$
{}_2\vec{w} \overset{\text{def}}{=} \begin{bmatrix} {}_2\vec{w}_3 \\ {}_2\vec{w}_2 \\ \vdots \\ {}_2\vec{w}_{m_2} \end{bmatrix} \in \mathbb{R}^{(n_2+1)\times m_2} \tag{14.2}
$$

$$
{}_2W \overset{\text{def}}{=} \begin{bmatrix} {}_2\vec{w}_3 & {}_2\vec{w}_2 & \cdots & {}_2\vec{w}_{m_2} \end{bmatrix} \tag{14.3}
$$

The linear term for Layer 2, which is the observation ${}_2\underline{u}$, will be written as

$$
\begin{aligned}
{}_2\underline{u} &\overset{\text{def}}{=} [{}_2\underline{x}\ 1]{}_2W \qquad (14.4)\\
&= {}_2\underline{x}[{}_2W_{n_2}] + 1[{}_2W_{n_2+1}]\\
{}_2\vec{u} &\overset{\text{def}}{=} {}_2\underline{u}^T \qquad (14.5)\\
&= [{}_2W]^T \begin{bmatrix} {}_2\vec{x} \\ 1 \end{bmatrix}
\end{aligned}
$$

Details of how we can use the Kronecker product to manage the linear term are provided in the extra notes for this class.

Layer 2 will have some uniform activation function ${}_2\phi(u)$; in practice, this activation function might be the logistic function or the Rectified Linear Unit (ReLU) function. The derivative of the activation function will be the function ${}_2\psi(u)$.

The output of Layer 2 will be m_2 scalar values, each being ${}_2\phi({}_2u_j)$. For the purpose of differentiation, we will gather these scalar values into a vector ${}_2\vec{\phi}$ that is a function of the linear vector ${}_2\vec{u}$, which we will write as

$$
\begin{aligned}
{}_2\vec{\phi} &\overset{\text{def}}{=} \begin{bmatrix} {}_2\phi({}_2u_3) \\ {}_2\phi({}_2u_2) \\ \vdots \\ {}_2\phi({}_2u_{m_2}) \end{bmatrix}\\
&= {}_2\vec{\phi}({}_2\vec{u}) \qquad (14.6)
\end{aligned}
$$

To feed these values forward to the output neuron, we will re-define the data parameter of the output neuron. Equation 12.15 will now become

$$
{}_3\underline{x}({}_2\vec{w}; \underline{x}) \overset{\text{def}}{=} [{}_2\vec{\phi}({}_2\vec{u}; \underline{x})]^T \tag{14.7}
$$

An important consequence of Equation 14.7 is that the data term ${}_3\underline{x}$ is changed from being a data *parameter* to being a *function* that has argument ${}_2\vec{w}$ and a data parameter $\underline{x}$.

In Equation 12.14, the objective function $f_3(\vec{w}; \underline{x})$ had a vector argument $\vec{w} = {}_3\vec{w}$ and a data parameter $\underline{x}$. When we add a layer, we will need a different objective function that has a different vector argument.

For the vector argument, we will partition the weight vector into the weights for Layer 3, which is ${}_3\vec{w}$, and the weights for Layer 2, which is ${}_2\vec{w}$. The combined weight vector is

$$\begin{aligned} \vec{w} &\overset{\text{def}}{=} \begin{bmatrix} {}_3\vec{w} \\ {}_2\vec{w} \end{bmatrix} \qquad (14.8) \\ \text{where} \quad {}_3\vec{w} &\in \mathbb{R}^{m_2+1} \\ {}_2\vec{w} &\in \mathbb{R}^{(n_2+1)m_2} \end{aligned}$$

We will write the 3-layer objective function as $f_3(\vec{w}; \underline{x})$. Our objective function, for a two-layer neural network, begins with terms from Equation 12.14 and continues with substitution of the new term for ${}_3\underline{x}$ of Equation 14.7.

Using terms from Equation 12.14, Equation 14.7, Equation 12.25, and Equation 12.24, the objective function for our 3-layer ANN is

$$\begin{aligned} f_3(r; y, \underline{x}) &\overset{\text{def}}{=} g(r; y, \underline{x}) \qquad (14.9) \\ g(r; y, \underline{x}) &\overset{\text{def}}{=} \frac{1}{2} r^T r \\ r({}_3\phi; y, \underline{x}) &\overset{\text{def}}{=} y - {}_3\phi(u; \underline{x}) \\ {}_3\phi({}_3u; \underline{x}) &\overset{\text{def}}{=} {}_3u(\vec{w}; \underline{x}) \\ {}_3u(\vec{w}; \underline{x}) &\overset{\text{def}}{=} \begin{bmatrix} {}_3\vec{x}^T & 1 \end{bmatrix} \begin{bmatrix} {}_3\vec{w} \end{bmatrix} = \begin{bmatrix} {}_3\vec{w}_{1..m_3-1} \end{bmatrix}^T {}_3\vec{x} + {}_3\vec{w}_{m_3} \\ {}_3\vec{x}({}_2\vec{w}; \underline{x}) &\overset{\text{def}}{=} {}_2\vec{\phi}({}_2\vec{w}; \underline{x}) \qquad (14.10) \\ {}_2\vec{\phi}({}_2\vec{u}; \underline{x}) &\overset{\text{def}}{=} \begin{bmatrix} {}_2\phi({}_2u_3; \underline{x}) \\ \vdots \\ {}_2\phi({}_2u_{m_2}; \underline{x}) \end{bmatrix} \\ {}_2\vec{u}({}_2\vec{w}; \underline{x}) &\overset{\text{def}}{=} \left[\begin{bmatrix} {}_2\vec{x}^T & 1 \end{bmatrix} {}_2W \right]^T \\ {}_2\vec{x}(; \underline{x}) &\overset{\text{def}}{=} \underline{x}^T \qquad (14.11) \end{aligned}$$

Consider the gradient of the objective function $f_3(\vec{w})$ in Equation 14.9. When we expand the gradient, we find that the first m_3 terms are with respect to the weight vector ${}_3\vec{w}$ of Layer 3 and that the next m_2 terms are with respect to the weight vector ${}_2\vec{w}$ of Layer 2. We will take the liberty of writing this gradient as

$$\begin{aligned}\underline{\nabla} f_3(\vec{w};\underline{x}) &\overset{\text{def}}{=} \begin{bmatrix}\frac{\partial f_3}{\partial w_3} & \frac{\partial f_3}{\partial w_2} & \cdots & \frac{\partial f_3}{\partial w_{m_3}} & \frac{\partial f_3}{\partial w_{m_3+1}} & \cdots & \frac{\partial f_3}{\partial w_{m_3+m_2}}\end{bmatrix} \\ &= \begin{bmatrix}\underline{\nabla} f_3({}_3\vec{w}) & \underline{\nabla} f_3({}_2\vec{w})\end{bmatrix} \qquad (14.12)\end{aligned}$$

We can now differentiate the activation for the second layer of this simple neural network. The three keys to understanding the differentiation, which will produce Equation 14.12, are:

1. The first m_3 terms of $\underline{\nabla} f_3(\vec{w})$ have only ${}_3\vec{w}$ as a vector argument
2. The next m_2 terms of $\underline{\nabla} f_3(\vec{w})$ have only ${}_2\vec{w}$ as a vector argument
3. These latter m_2 terms can be computed separately from the first m_3 terms

When we differentiate Equation 14.9, we will need to understand two terms in detail. The term ${}_2\vec{\phi}({}_2\vec{u};\underline{x})$ is a vector function with a vector argument, so its derivative is a Jacobian matrix. The i^{th} entry of ${}_2\vec{\phi}({}_2\vec{u})$ depends on variable ${}_2u_i$ and not on any other entry of the linear term ${}_2\vec{u}$; this implies that

$$\frac{\partial\, {}_2\vec{\phi}_i}{\partial {}_2u_{j\neq i}} = 0 \qquad (14.13)$$

From Equation 14.13, the Jacobian matrix ${}_2J$ must be a diagonal matrix. We can write the diagonal entries as a vector and concisely represent the Jacobian matrix as

$$\begin{aligned}{}_2\vec{\psi} &\overset{\text{def}}{=} \frac{\partial\, {}_2\vec{\phi}_i}{\partial {}_2u_i} && (14.14)\\ {}_2J &\overset{\text{def}}{=} \text{diag}({}_2\vec{\psi}) && (14.15)\\ &= {}_2J^T && (14.16)\end{aligned}$$

We can differentiate Equation 14.9, using terms from Equation 14.15 and Equation 14.34, as

$$
\begin{aligned}
\frac{\partial f_3}{\partial g} &= 1 \\
\frac{\partial g}{\partial r} &= r \\
\frac{\partial r}{\partial\, {}_3\phi} &= -1 \\
\frac{\partial\, {}_3\phi}{\partial\, {}_3u} &= {}_3\psi = 1 \\
\frac{\partial\, {}_3u}{\partial\, {}_3\vec{w}} &= \begin{bmatrix} {}_3\vec{x}^T & 1 \end{bmatrix} \\
\frac{\partial\, {}_3u}{\partial\, {}_3\vec{x}} &= {}_3\vec{w}^T_{1..n_3}\ {}_3b \\
\frac{\partial\, {}_3\vec{x}}{\partial\, {}_2\vec{\phi}} &= I \\
\frac{\partial\, {}_2\vec{\phi}}{\partial\, {}_2\vec{u}} &= {}_2J \\
\frac{\partial\, {}_2\vec{u}}{\partial\, {}_2\vec{w}} &= \begin{bmatrix} I \otimes [{}_2\vec{x}^T\ 1] \end{bmatrix}
\end{aligned}
\qquad (14.17)
$$

We can write the gradient $\underline{\nabla} f_3(\vec{w}; \underline{x})$ by using the block partitioning of Equation 14.12. The first 5 terms of Equation 14.17 are gathered into $\underline{\nabla} f_3({}_3\vec{w})$ and the last 4 terms of Equation 14.17 are gathered into $\underline{\nabla} f_3({}_2\vec{w})$. For Layer 3, the gradient term and the direction of steepest descent are much as they were for the objective function $f_3(g)$, being

$$
\begin{aligned}
\underline{\nabla} f_3({}_3\vec{w}; \underline{x}) &= \frac{\partial f_3}{\partial g}\frac{\partial g}{\partial r}\frac{\partial r}{\partial\, {}_3\phi}\frac{\partial\, {}_3\phi}{\partial\, {}_3u}\frac{\partial\, {}_3u}{\partial\, {}_3\vec{w}} \\
&= (1)(r)(-1)(1)[{}_3\vec{x}^T\ 1] \\
{}_3\vec{d} &= -\left[\frac{\partial\, {}_3u}{\partial\, {}_3\vec{w}}\right]^T \left[\frac{\partial\, {}_3\phi}{\partial\, {}_3u}\right]^T \left[\frac{\partial\, r}{\partial\, {}_3\phi}\right]^T \left[\frac{\partial g}{\partial r}\right]^T \left[\frac{\partial f_3}{\partial g}\right]^T \\
&= -\left[\begin{bmatrix} {}_3\vec{x} \\ 1 \end{bmatrix} (1)\right]((-1)(r)(1)) \\
&= -[{}_3\vec{s}]_3 b
\end{aligned}
\qquad (14.18)
$$

For Layer 2, using the substitution of Equation 14.34 and the identity ${}_2J^T = {}_2J$, the gradient term and the direction of steepest descent are

$$\begin{aligned}
\underline{\nabla} f_3({}_2\vec{w};\underline{x}) &= \frac{\partial\,{}_3u}{\partial\,{}_3\vec{x}}\frac{\partial\,{}_2\vec{\phi}}{\partial\,{}_2\vec{u}}\frac{\partial\,{}_2\vec{u}}{\partial\,{}_2\vec{w}} \qquad (14.19)\\
&= {}_3\vec{w}^T_{1..n_3} I[{}_2J]\left[I\otimes[{}_2\vec{x}^T\ 1]\right]\\
&= {}_3\vec{w}^T_{1..n_3}[{}_2J]\left[I\otimes[{}_2\vec{x}^T\ 1]\right]\\
{}_2\vec{d} &= -[\underline{\nabla} f_3({}_2\vec{w};\underline{x})]^T\\
&= -\left[I\otimes\begin{bmatrix}{}_2\vec{x}\\1\end{bmatrix}\right][{}_2J]{}_3\vec{w}_{1..n_3}\ {}_3b \qquad (14.20)
\end{aligned}$$

The term ${}_3\vec{w}_{1..n_3}$ in Equation 14.20 is the updated term from Layer 3 of our neural network. This new term is the back-propagation term for Layer 2; we will abbreviate this term as ${}_2\vec{b}$ and write the descent vector for Layer 2 as

$$\begin{aligned}
{}_2\vec{b} &\stackrel{\text{def}}{=} {}_3\vec{w}_{1..n_3}\ {}_3b \qquad (14.21)\\
{}_2\vec{d} &= -\left[I\otimes\begin{bmatrix}{}_2\vec{x}\\1\end{bmatrix}\right][{}_2J]{}_2\vec{b} \qquad (14.22)
\end{aligned}$$

The expression for ${}_2\vec{d}$ in Equation 14.22 involves sparse matrices. The computation of the descent vector can be simplified using the Hadamard product; derivation of a simpler computation is provided in the extra notes for this class. When we use the simplifications of Equation 14.38 in the extra notes, we find that we can write the vector of steepest descent as

$$\begin{aligned}
{}_2S &\stackrel{\text{def}}{=} \begin{bmatrix}{}_2\vec{x}\\1\end{bmatrix}\left[{}_2\vec{\psi}({}_2\vec{u})\odot{}_2\vec{b}\right]^T \qquad (14.23)\\
{}_2\vec{d} &= -\text{vec}({}_2S) \qquad (14.24)
\end{aligned}$$

For the objective function $f_3(g)$ that describes the output of the 3-layer neural network, where all of the neuron weights are gathered into a single argument vector $\vec{w}$, the direction of steepest descent $\vec{d}$ is

$$\vec{d} \stackrel{\text{def}}{=} \begin{bmatrix}{}_3\vec{d}\\{}_2\vec{d}\end{bmatrix} \qquad (14.25)$$

14.2 ANN Layer 3 – Summary and Computations

We can rephrase the previous observations on the relationships of outputs and inputs to weight vectors, in mathematical notation, as:

	Input	**Output(s)**	**No. of Neurons**
Layer 1	${}_1\underline{x} = \underline{x}$	$\vec{x}$	$n+1$
Layer 2	${}_2\underline{x} = [{}_1\underline{x}\ \ 1]$	${}_2\phi_j({}_2\vec{w})$	m_2
Layer 3	${}_3\underline{x}({}_2\vec{w};\underline{x})$	${}_3\phi({}_3\vec{w},{}_2\vec{w})$	1

For a single data observation $\underline{x}$, our 3-layer neural network would be processed in a two-pass algorithm:

```
Pass 1: Forward Evaluation of $\phi$ Terms
    Layer 1:
        Set ${}_2\underline{x} = [{}_1\underline{x}\ \ 1]$
    Layer 2:
        Compute ${}_2\vec{\phi}$ using ${}_2\vec{w}$ and ${}_2\underline{x}$
        Compute ${}_2\vec{\psi}$
        Compute ${}_3\underline{x}$
    Layer 3:
        Compute ${}_3\phi$ using ${}_3\vec{w}$ and ${}_3\underline{x}$
        Compute ${}_3\psi$
    Output:
        Compute $r$ using ${}_3\phi$ and $y$
        Compute $f_3(g(r))$

Pass 2: Back Propagation of $\vec{d}$ Terms
    Initialize:
        Compute $b_3$
    Layer 3:
        Compute ${}_3\vec{s}$ and ${}_3\vec{d}$
        Set ${}_2\vec{b} \leftarrow [{}_3\vec{w}_{1..n_3}]\ {}_3\psi_3 b$
        Set $\vec{d} \leftarrow {}_3\vec{d}$
    Layer 2:
        Compute ${}_2\vec{d}$ and ${}_1\vec{b}$
        Set ${}_2S \leftarrow {}_2\underline{x}\left[{}_2\vec{\psi}({}_2\vec{u}) \odot {}_2\vec{b}\right]^T$
        Set ${}_2\vec{d} \leftarrow -\text{vec}({}_2S)$
        Set $\vec{d} \leftarrow \begin{bmatrix} \vec{d} \\ {}_2\vec{d} \end{bmatrix}$
        Set ${}_1\vec{b} \leftarrow \text{vec}\left({}_2W \odot \left[{}_2\vec{\psi} \odot {}_2\vec{b}\right]^T\right)$
```

14.3 ANN – Layer 4 and Beyond

The methods of Section 14.1 can be extended to any number $p \geq 4$ layers. The feed-forward computations of Section 14.2 will have, for each additional layer $\#l$, a weight matrix ${}_lW$ and an activation function ${}_l\phi(u)$. The back-propagation computations will use the same process as for Layer 3, back-propagating scale factors and using the input of the layer as the unscaled gradient. Back-propagation must be done with special attention because layers become function with vectorial inputs and results.

Handling multiple data vectors $\vec{x}_j$, each providing a descent vector $\vec{d}$, can be implemented in many ways. Two common ways are to perform the computations for each data vector, or to parallelize the computations to allow for a design matrix X. Such extensions are beyond the scope of this course.

14.4 Example: The 2D "Exclusive-Or" Problem

A frequently used example in neural networks is often called the *exclusive-or* problem. A neuron that uses a hyperplane decision process cannot solve certain simple problems. Suppose that a 2D vector $\vec{v} \in \mathbb{R}^2$. We can define an exclusive-or function on $\vec{v}$ that is 1 if $\vec{v}$ is in Quadrant 1 or Quadrant 3 of the plane, and is 0 otherwise, as

$$g(\vec{v}) \overset{\text{def}}{=} \begin{cases} 1 & \text{if} \quad \text{sign}(v_1)\text{sign}(v_2) > 0 \\ 0 & \quad\ \ \text{otherwise} \end{cases} \tag{14.26}$$

Example data that are labeled according to Equation 14.26, and a simple neural network that has $m = 2$ neurons in the hidden layer, are shown in Figure 14.2.

We can construct a single layer of $m = 2$ neurons for this problem, using the computations in Section 14.2. Suppose that we use the method of steepest descent. Using a stepsize $s = 0.001$, the

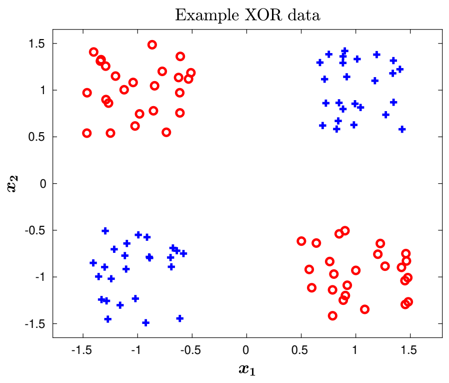

Figure 14.2: Data for a 2D "exclusive or" problem. Vectors in Quadrant 1 and Quadrant 3, given as Label 1, are shown as blue "plus" signs; vectors in Quadrant 2 and Quadrant 4, given as Label 0, are shown as red circles.

initial weight vector $\vec{w}_0$ and the estimated minimizer after 980 iterations are

$$\vec{w}_0 = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \end{bmatrix} \qquad \vec{w}^* \approx \begin{bmatrix} 0.4705 \\ -0.3254 \\ 0.0100 \\ 0.4662 \\ 0.5786 \\ -0.2181 \\ 0.9626 \\ 1.0986 \\ 1.0711 \end{bmatrix} \tag{14.27}$$

Ordinarily, a weight vector such as $\vec{w}^*$ in Equation 14.27 is difficult for a human to interpret. In this case, we know that ${}_3\vec{w}_3$ is the decision hyperplane for the first neuron in the hidden layer, and that ${}_3\vec{w}_2$ is the decision hyperplane for the second neuron in the hidden layer. These weight vectors are

$${}_2\vec{w}_3^* \approx \begin{bmatrix} 0.4662 \\ 0.5786 \\ -0.2181 \end{bmatrix} \qquad {}_2\vec{w}_2^* \approx \begin{bmatrix} 0.9626 \\ 1.0986 \\ 1.0711 \end{bmatrix} \tag{14.28}$$

We can plot these hyperplanes along with the data, using the weight vector of Equation 14.27 to classify the data. As shown in Figure 14.3, the classifications are correct and the $f_3(\vec{w}) = 0$ contours approximately divide the positive and negative labels. The weight vector for the output neuron performs the final classification, using the sigmoid outputs of the hidden neurons in a linear decision process.

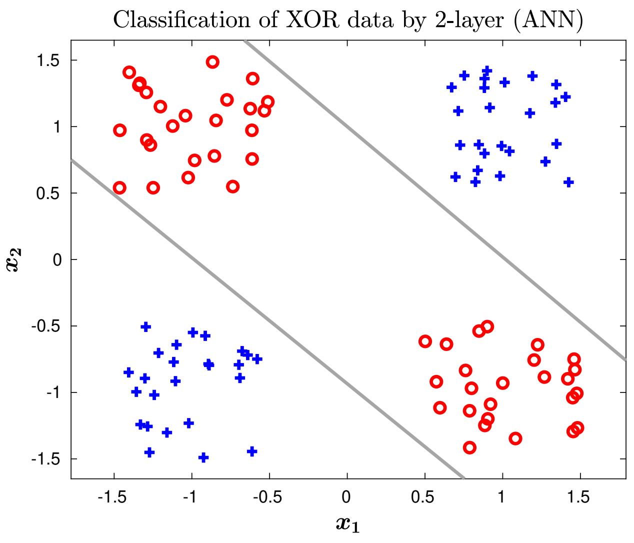

Figure 14.3: Vectors in 2D that are classified using a simple 3-layer neural network. Vectors with Label 1 are shown as blue "plus" signs. Vectors with Label 0 are shown as red circles. The black lines are the $f(\vec{w}) = 0$ contours of the outputs of the two neurons in the hidden layer; each contour is a level curve of a sigmoid activation function.

Because each weight vector has 3 entries, we can plot the weight vectors to visualize the method of steepest descent. Using the above arguments, Figure 14.4 shows the "tracks" of the weight vector as the iteration proceeds.

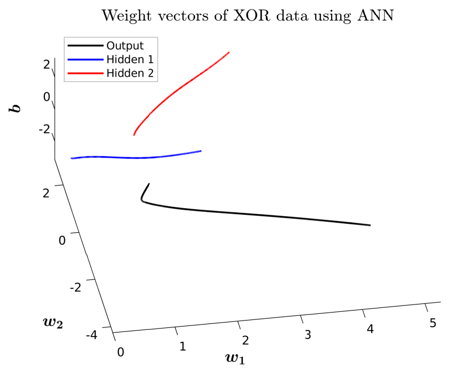

Figure 14.4: Three "tracks" of the weight vectors during iterations of steepest descent. The output weights are shown in black. The weights of the hidden neurons are shown in blue and red. The hidden weights do not exhibit substantial changes and the output weights display the effects of using a Heaviside step function to classify the output of the neural network.

14.5 Extra Notes on Layer 2 Derivations

There are many technical details to the derivation of the equations for Layer 2 of our neural networks.

14.5.1 Weight Matrix – ${}_2W$

The weights Layer 2 are

$$\begin{aligned}
{}_2W &\overset{\text{def}}{=} \begin{bmatrix} {}_2\vec{w}_3 & {}_2\vec{w}_2 & \cdots & {}_2\vec{w}_{m_2} \end{bmatrix} \\
{}_2W_{n_2} &\overset{\text{def}}{=} \begin{bmatrix} {}_2\vec{w}_{1,1..n_2} & {}_2\vec{w}_{2,1..n_2} & \cdots & {}_2\vec{w}_{m_2,1..n_2} \end{bmatrix} \in \mathbb{R}^{(n_2)\times m_2} && (14.29) \\
{}_2W_{n_2+1} &\overset{\text{def}}{=} \begin{bmatrix} {}_2\vec{w}_{1,n_2+1} & {}_2\vec{w}_{2,n_2+1} & \cdots & {}_2\vec{w}_{m_2,n_2+1} \end{bmatrix} \in \mathbb{R}^{1\times m_2} && (14.30) \\
{}_2W &= \begin{bmatrix} {}_2W_{n_2} \\ {}_2W_{n_2+1} \end{bmatrix} && (14.31)
\end{aligned}$$

In Equation 14.31, the last row of the matrix ${}_2W$ are the values of the bias scalars for the neuron in Layer 2.

14.5.2 Linear Term – ${}_2\underline{u}$

The linear term for Layer 2, from Equation 14.4, can be written as

$$\begin{aligned}
{}_2\underline{u} &\overset{\text{def}}{=} [{}_2\underline{x}\ 1]{}_2W \\
&= {}_2\underline{x}[{}_2W_{n_2}] + 1[{}_2W_{n_2+1}] \\
{}_2\vec{u} &\overset{\text{def}}{=} {}_2\underline{u}^T \\
&= [{}_2W]^T \begin{bmatrix} {}_2\vec{x} \\ 1 \end{bmatrix} \\
&= [{}_2W_{n_2}]^T {}_2\vec{x} + [{}_2W_{n_2+1}]
\end{aligned}$$

We can also write the term ${}_2\vec{u}$ of Equation 14.5 by applying the Kronecker-vectorization theorem of Theorem 11.15 to Equation 14.4 to find that

$$\begin{array}{rrcl} & {}_2\underline{u} & = & [{}_2\underline{x}\ 1]{}_2W \\ \equiv & {}_2\underline{u} & = & [{}_2\vec{x}^T\ 1]{}_2W \\ \equiv & \mathrm{vec}({}_l\,\underline{u}) & = & \big[I \otimes [{}_2\vec{x}^T\ 1]\big]\ \mathrm{vec}({}_2W) \\ \equiv & {}_2\vec{u} & = & \big[I \otimes [{}_2\vec{x}^T\ 1]\big]\ {}_2\vec{w} \end{array} \tag{14.32}$$

The derivative of the term ${}_2\vec{u}({}_2\vec{w};\underline{x})$, using Equation 14.32, is

$$\frac{\partial\,{}_2\vec{u}}{\partial\,{}_2\vec{w}} \overset{\text{def}}{=} \big[I \otimes [{}_2\vec{x}^T\ 1]\big] \tag{14.33}$$

Using a property of the Kronecker product, we can write the transpose of Equation 14.33 as

$$\begin{array}{rcl} \big[I \otimes [{}_2\vec{x}^T\ 1]\big]^T & = & I^T \otimes [{}_2\vec{x}^T\ 1]^T \\ & = & I \otimes \begin{bmatrix} {}_2\vec{x} \\ 1 \end{bmatrix} \\ \left[\dfrac{\partial_2\,\vec{u}}{\partial_2\vec{w}}\right]^T & = & I \otimes \begin{bmatrix} {}_2\vec{x} \\ 1 \end{bmatrix} \end{array} \tag{14.34}$$

14.5.3 Steepest-Direction Vector – ${}_2\vec{s}$

When we expand the rightmost product in Equation 14.22, we see that

$$\begin{array}{rcl} [{}_2J]{}_2\vec{b} & = & \begin{bmatrix} {}_2\psi({}_2u_3) & & & \\ & {}_2\psi({}_2u_2) & & \\ & & \ddots & \\ & & & {}_2\psi({}_2u_{n_3}) \end{bmatrix} \begin{bmatrix} {}_2b_1 \\ {}_2b_2 \\ \vdots \\ {}_2b_{n_3} \end{bmatrix} \\ & = & {}_2\vec{\psi}({}_2\vec{u}) \odot {}_3\vec{b} \\ {}_2\vec{d} & = & -\left[I \otimes \begin{bmatrix} {}_2\vec{x} \\ 1 \end{bmatrix}\right] \left[{}_2\vec{\psi}({}_2\vec{u}) \odot {}_3\vec{b}\right] \end{array} \tag{14.35}$$

We see that Equation 14.35 contains a mixed Kronecker matrix-vector product. The mixed product in the right-hand term can be rewritten by using the substitution of Theorem 11.15 to be

$$
\begin{aligned}
{}_2S &\overset{\text{def}}{=} \begin{bmatrix} {}_2\vec{x} \\ 1 \end{bmatrix} \begin{bmatrix} {}_2\vec{\psi}({}_2\vec{u}) \odot {}_3\vec{b} \end{bmatrix}^T && (14.36) \\
&= \begin{bmatrix} {}_2\vec{s}_3 & {}_2\vec{s}_3 & \cdots & {}_2\vec{s}_{n3} \end{bmatrix} \\
{}_2\vec{s}_j &= \begin{bmatrix} {}_2\vec{x} \\ 1 \end{bmatrix} {}_2\psi({}_2u_j) b_{2,j} && (14.37) \\
{}_2\vec{d} &= -\left[I \otimes \begin{bmatrix} {}_2\vec{x} \\ 1 \end{bmatrix} \right] \begin{bmatrix} {}_2\vec{\psi}({}_2\vec{u}) \odot {}_3\vec{b} \end{bmatrix} \\
&= -\text{vec}({}_2S) && (14.38)
\end{aligned}
$$

The matrix ${}_2S$ of Equation 14.36 has columns that are the steepest vectors for the neurons in Layer 2, i.e., ${}_2\vec{s}_j$ is the steepest vector for the j^{th} neuron in Layer 2; Equation 14.37 explicitly describes the j^{th} such vector. Vectorizing th matrix ${}_2S$produces the vector of steepest descent ${}_2\vec{d}$ for the entire layer. Computations might be performed using the matrix ${}_2S$ or using the gathered vector ${}_2\vec{d}$, depending on the implementation.

End of Extra Notes

References

[1] Rumelhart DE, Hinton GE, Williams RJ: Learning representations by back-propagating errors. *Nature* 323(6088):533–536, 1986