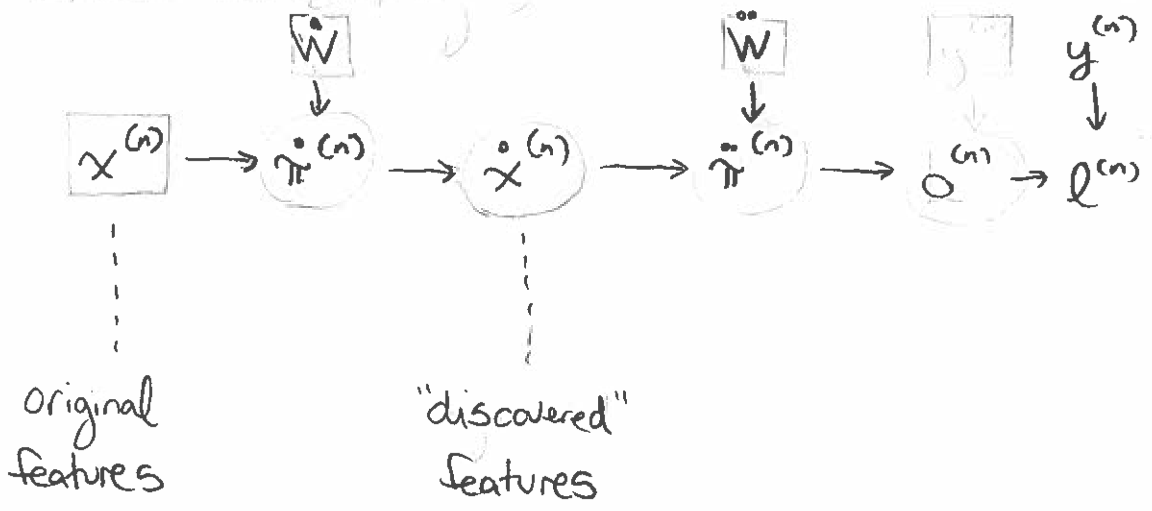
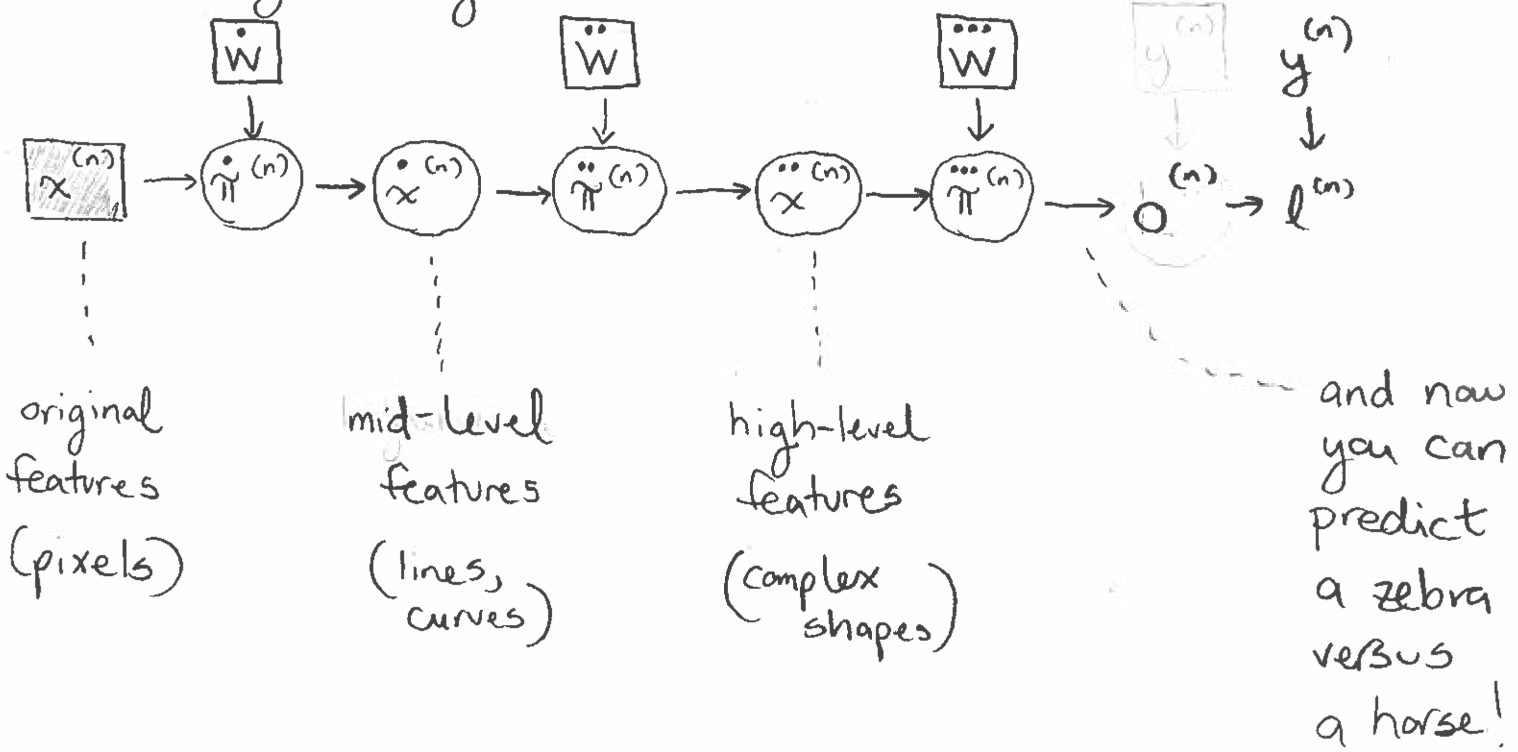
NEURAL NETWORKS AND BACKPROPAGATION

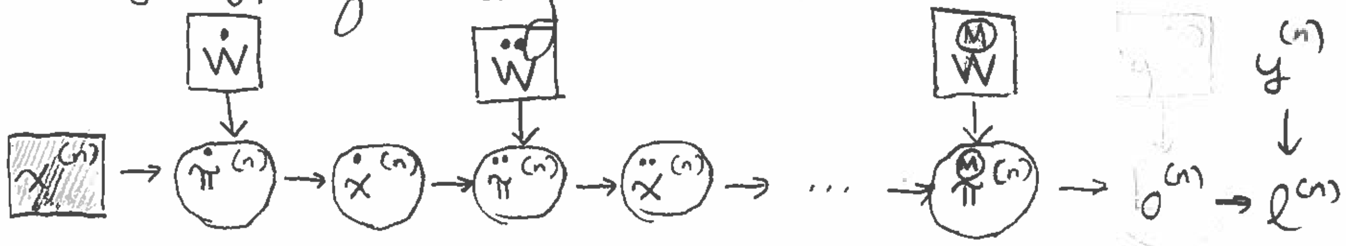
① Consider again our feature discovery network:



② We could consider generalizing this model to provide multiple layers of feature discovery, e.g. for image recognition:

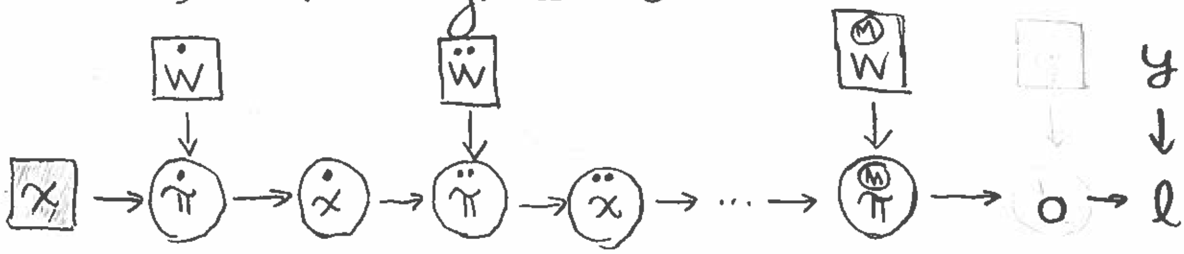


(3) In its full generality:

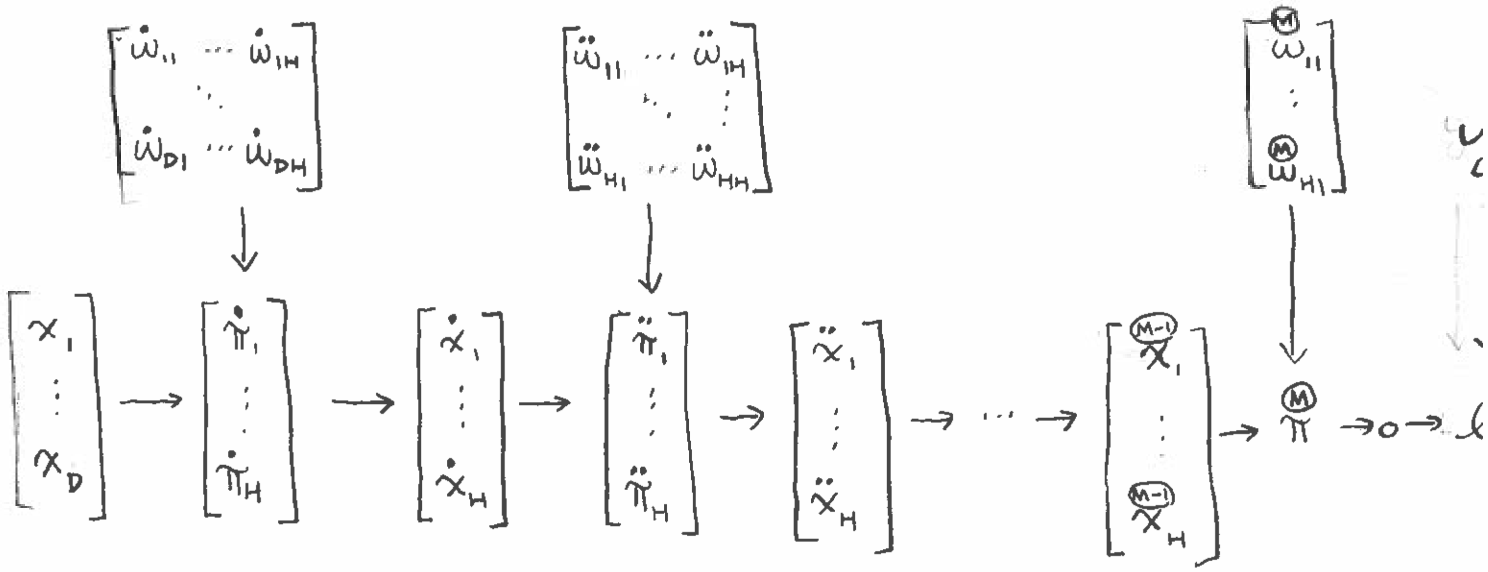


this is called an <u>M-layer feedforward neural network</u>.

Let's drop all those (n) superscripts for convenience (we'll bring them back when needed to avoid confusion). This gives us:



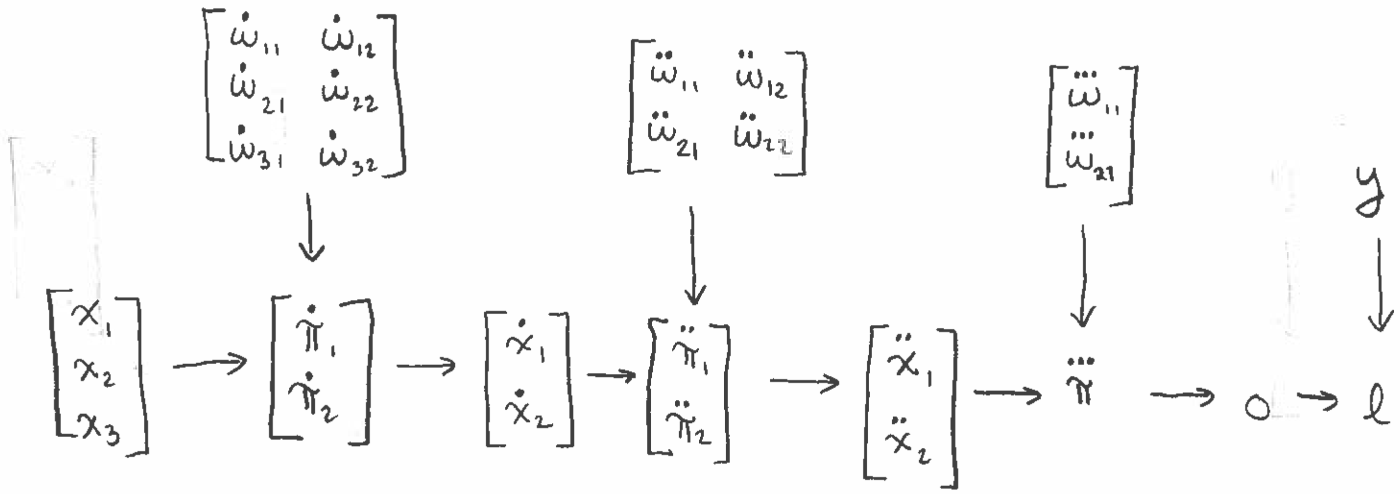
Just in case we've forgotten which of these are vectors and which are matrices, here it is explicitly:



We assume each "feature discovery" layer discovers H features.

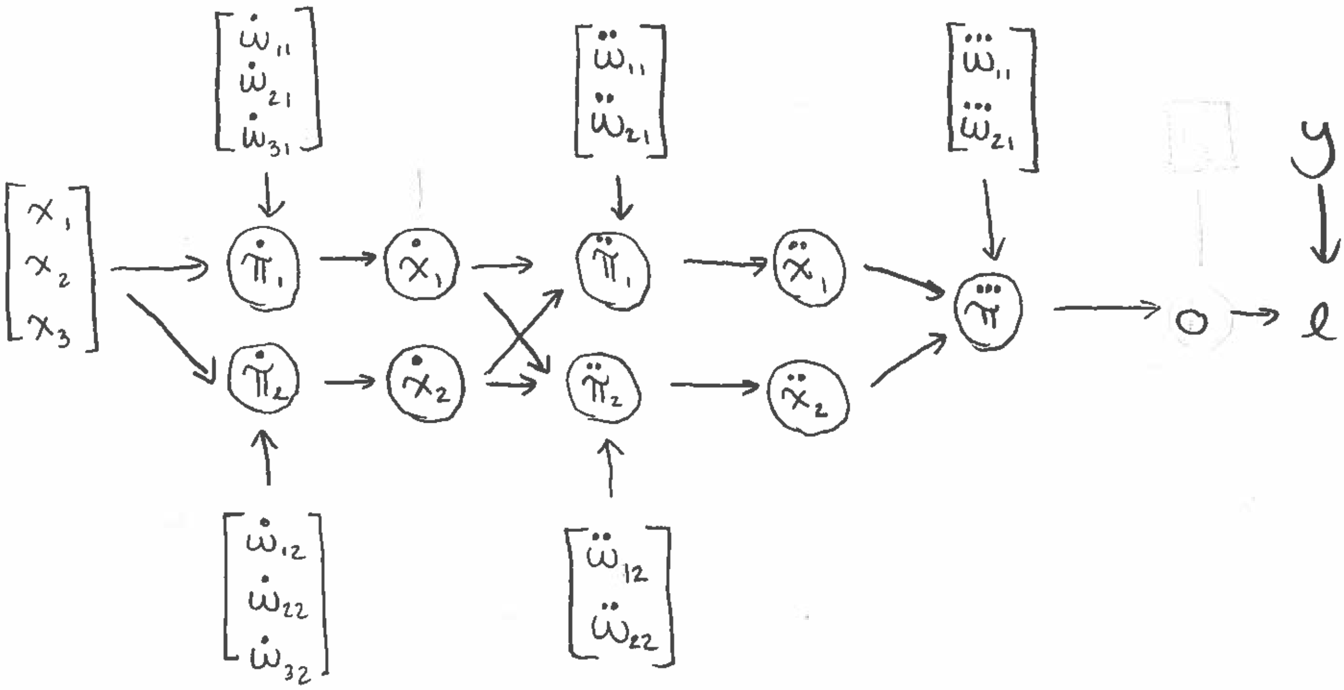
④ To train this model using gradient descent, we need to be able to compute $\dfrac{\partial l}{\partial \overset{m}{w}_{ij}}$ for each weight $\overset{m}{w}_{ij}$.

Before doing this in its full generality, let's see how we can compute these derivatives for a 3-layer network where $H=2$ and $D=3$.



Neural Networks and Backpropagation

(5) As we did before for the feature discovery network, let's break down the endogenous variables into scalars to make it easier to apply the Chain Rule of Partial Derivatives:



(6) Our goal is to compute (for all relevant i,j):

$$\frac{\partial \ell}{\partial \dot{\omega}_{ij}} \quad \text{and} \quad \frac{\partial \ell}{\partial \ddot{\omega}_{ij}} \quad \text{and} \quad \frac{\partial \ell}{\partial \dddot{\omega}_{ij}}$$

First, we can observe that $\dddot{\pi}$ separates ℓ from all $\overset{(m)}{\omega}_{ij}$, so:

$$\frac{\partial \ell}{\partial \overset{(m)}{\omega}_{ij}} = \frac{\partial \ell}{\partial \dddot{\pi}} \cdot \frac{\partial \dddot{\pi}}{\partial \overset{(m)}{\omega}_{ij}}$$

This is the just the standard derivative of the loss function.

NEURAL NETWORKS AND BACKPROPAGATION

(7) So the challenge is to compute $\dfrac{\partial \dddot{\pi}}{\partial \overset{(m)}{\omega}_{ij}}$ for any layer m.

It's straightforward for $m=3$:

$$\frac{\partial \dddot{\pi}}{\partial \dddot{\omega}_{ij}} = \frac{\partial}{\partial \dddot{\omega}_{ij}}\left(\begin{bmatrix}\ddot{x}_1 & \ddot{x}_2\end{bmatrix}\begin{bmatrix}\dddot{\omega}_{11}\\ \dddot{\omega}_{21}\end{bmatrix}\right) = \ddot{x}_i$$

---

(8) What about $m=2$?

$$\frac{\partial \dddot{\pi}}{\partial \ddot{\omega}_{ij}} = \frac{\partial \dddot{\pi}}{\partial \ddot{\pi}_j} \cdot \frac{\partial \ddot{\pi}_j}{\partial \ddot{\omega}_{ij}}$$

[$\ddot{\pi}_j$ separates $\dddot{\pi}$ from $\ddot{\omega}_{ij}$, so Chain Rule applies]

$$= \frac{\partial \dddot{\pi}}{\partial \ddot{\pi}_j} \cdot \dot{x}_i$$

$$\left[\frac{\partial \ddot{\pi}_j}{\partial \ddot{\omega}_{ij}} = \dot{x}_i\right]$$

---

(9) What about $m=1$?

$$\frac{\partial \dddot{\pi}}{\partial \dot{\omega}_{ij}} = \frac{\partial \dddot{\pi}}{\partial \dot{\pi}_j} \frac{\partial \dot{\pi}_j}{\partial \dot{\omega}_{ij}}$$

[$\dot{\pi}_j$ separates $\dddot{\pi}$ from $\dot{\omega}_{ij}$, so Chain Rule applies]

$$= \frac{\partial \dddot{\pi}}{\partial \dot{\pi}_j} \cdot x_i$$

$$\left[\frac{\partial \dot{\pi}_j}{\partial \dot{\omega}_{ij}} = x_i\right]$$

NEURAL NETWORKS AND BACKPROPAGATION

(10) In summary:

$$\left.\begin{aligned}
\frac{\partial \dddot{\pi}}{\partial \dddot{w}_{ij}} &= \ddot{x}_i \\
\frac{\partial \dddot{\pi}}{\partial \ddot{w}_{ij}} &= \dot{x}_i \frac{\partial \dddot{\pi}}{\partial \ddot{\pi}_j} \\
\frac{\partial \dddot{\pi}}{\partial \dot{w}_{ij}} &= x_i \frac{\partial \dddot{\pi}}{\partial \dot{\pi}_j}
\end{aligned}\right\}$$

For the general case:

$$\frac{\partial \overset{(M)}{\pi}}{\partial \overset{(m)}{w}_{ij}} = \overset{(m-1)}{x}_i \cdot \underbrace{\frac{\partial \overset{(M)}{\pi}}{\partial \overset{(m)}{\pi}_j}}$$

so how do we compute this term?

(11) Consider $\frac{\partial \dddot{\pi}}{\partial \dot{\pi}_j}$ for our 3-layer network.

$$\frac{\partial \dddot{\pi}}{\partial \dot{\pi}_j} = \frac{\partial \dddot{\pi}}{\partial \dot{x}_j} \frac{\partial \dot{x}_j}{\partial \dot{\pi}_j}$$

[$\dot{x}_j$ separates $\dddot{\pi}$ from $\dot{\pi}_j$, so Chain Rule applies]

$$= \left(\sum_{h=1}^{2} \frac{\partial \dddot{\pi}}{\partial \ddot{\pi}_h} \frac{\partial \ddot{\pi}_h}{\partial \dot{x}_j} \right) \frac{\partial \dot{x}_j}{\partial \dot{\pi}_j}$$

[$\{\ddot{\pi}_1, \ddot{\pi}_2\}$ separates $\dddot{\pi}$ from $\dot{x}_j$ so Chain Rule applies]

$$= \frac{\partial \dot{x}_j}{\partial \dot{\pi}_j} \sum_{h=1}^{2} \frac{\partial \ddot{\pi}_h}{\partial \dot{x}_j} \frac{\partial \dddot{\pi}}{\partial \ddot{\pi}_h}$$

$$= a'(\dot{\pi}_j) \sum_{h=1}^{2} \ddot{w}_{hj} \underbrace{\frac{\partial \dddot{\pi}}{\partial \ddot{\pi}_h}}$$

but this can be computed recursively in the same way!

(12) In summary (for our 3-layer example):

$$\frac{\partial \dddot{\pi}}{\partial \dot{\pi}_j} = a'(\dot{\pi}_j) \sum_{h=1}^{2} \ddot{\omega}_{hj} \frac{\partial \dddot{\pi}}{\partial \ddot{\pi}_h}$$

and for the general case:

$$\frac{\partial \overset{(M)}{\pi}}{\partial \overset{(M)}{\pi}} = \boxed{}$$ ← base case

$$\frac{\partial \overset{(M)}{\pi}}{\partial \overset{(m)}{\pi}_j} = \boxed{}$$ ← recursive step

(13) Putting it all together, we have cobbled together a strategy for computing every partial derivative $\frac{\partial L}{\partial \overset{(m)}{\omega}_{ij}}$:

BACKPROPAGATION:

(a) for m in range($\boxed{}$) and j in range($\boxed{}$)

compute $\frac{\partial \overset{(M)}{\pi}}{\partial \overset{(m)}{\pi}_j} = \boxed{}$

(b) $\frac{\partial L}{\partial \overset{(m)}{\omega}_{ij}} = \boxed{} \cdot \frac{\partial \overset{(M)}{\pi}}{\partial \overset{(m)}{\pi}_j}$

Because we compute the partial derivatives $\frac{\partial \overset{(M)}{\pi}}{\partial \overset{(m)}{\pi}_j}$ starting from the final layer M and moving back, we call it backpropagati

NEURAL NETWORKS AND BACKPROPAGATION

(14) Because we compute the partial derivatives $\dfrac{\partial \pi^{(M)}}{\partial \pi^{(m)}}$ starting from the final layer M and moving backwards, this algorithm is known as <u>backpropagation</u>.