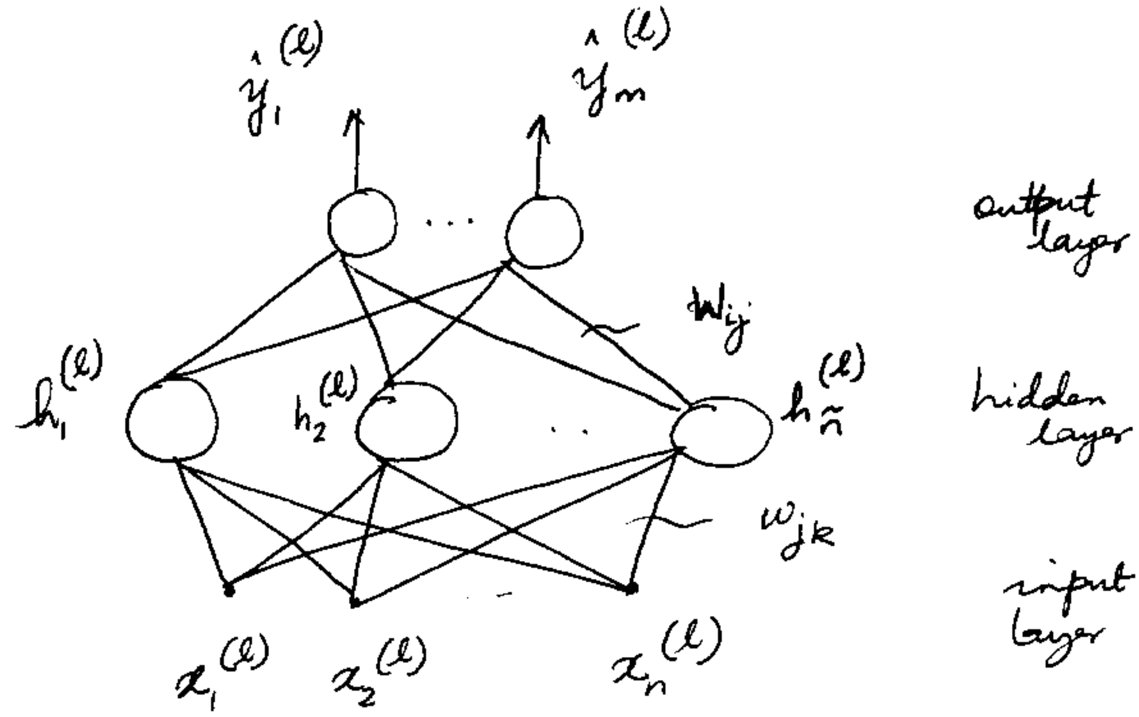
# Back-Propagation (The Generalized Delta Rule)




The bias is present though not shown explictly.

BP is a supervised learning algorithm.

Training data is $\left\{ \left( x^{(l)}, y^{(l)} \right) \right\}_{l=1}^{N}$

---

Note: we will refer to the outputs using the index $i$
. . . . hidden layer neurons - - - $j$
. . . inputs . . - - $k$

so, when we say $W_{ij}$ it implies the weight between output $i$ and hid. neu. $j$

- - — $W_{jk}$ . . . hidden neuron $j$ and input $k$.

- The hidden neuron receives as net input

$$s_j^{(\ell)} = \sum_{k=0}^{n} w_{jk}\, x_k^{(\ell)} \quad \cdots (1)$$

O/P of hidden layer neuron is :-

$$h_j^{(\ell)} = f\left(s_j^{(\ell)}\right) = f\left[\sum_{k=0}^{n} w_{jk}\, x_k^{(\ell)}\right] \quad \cdots (2)$$

- An output layer neuron receives as net input

$$S_i^{(\ell)} = \sum_{j=0}^{\hat{n}} W_{ij}\, h_j^{(\ell)} \quad \cdots (3)$$

note: $h_0^{(\ell)} = 1$ the bias input

O/P of output layer neuron is :-

$$\hat{y}_i^{(\ell)} = f\left(S_i^{(\ell)}\right) = f\left[\sum_{j=0}^{\hat{n}} W_{ij}\, h_j^{(\ell)}\right] \quad \cdots (4)$$

Take an error (loss) function as before,

$$J^{(\ell)} = \frac{1}{2} \sum_{i=1}^{m} \left(y_i^{(\ell)} - \hat{y}_i^{(\ell)}\right)^2 \quad \cdots (5)$$

$$= \frac{1}{2} \sum_{i=1}^{m} e_i^{(\ell)\,2} \quad \cdots (6)$$

$$J = \sum_{\ell=1}^{N} J^{(\ell)}$$

Now we can use gradient descent to find the update for the weights, i.e.

$$W_{ij} = W_{ij} - \eta \frac{\partial J^{(\ell)}}{\partial W_{ij}}$$

as before we are using pattern-wise or online update

what is $\dfrac{\partial J^{(\ell)}}{\partial W_{ij}}$

$$\frac{\partial J^{(\ell)}}{\partial W_{ij}} = \frac{\partial J^{(\ell)}}{\partial e_i^{(\ell)}} \cdot \frac{\partial e_i^{(\ell)}}{\partial \hat{y}_i^{(\ell)}} \cdot \frac{\partial \hat{y}_i^{(\ell)}}{\partial s_i^{(\ell)}} \cdot \frac{\partial s_i^{(\ell)}}{\partial W_{ij}}$$

$$= e_i^{(\ell)} \cdot (-1) \cdot f'(s_i^{(\ell)}) \cdot h_j^{(\ell)}$$

$$= - e_i^{(\ell)} \cdot f'(s_i^{(\ell)}) \cdot h_j^{(\ell)}$$

so,

$$W_{ij} = W_{ij} + \Delta W_{ij}$$

$$= W_{ij} - \eta \frac{\partial J^{(\ell)}}{\partial W_{ij}}$$

$$= W_{ij} + \eta \cdot e_i^{(\ell)} \cdot f'(s_i^{(\ell)}) \cdot h_j^{(\ell)}$$

$$W_{ij} = W_{ij} + \eta \underbrace{\left[ y_i^{(\ell)} - \hat{y}_i^{(\ell)} \right] f'(s_i^{(\ell)})}_{d_i^{(\ell)}} \cdot h_j^{(\ell)} \quad \ldots (7)$$

We now know how to update the o/p to hidden weights. To get the hidden to input weights,

Recall eqn. 5 and 6

$$J^{(l)} = \frac{1}{2} \sum_{i=1}^{m} e_i^{(l)^2}$$

$$= \frac{1}{2} \sum_{i=1}^{m} \left[ y_i^{(l)} - \hat{y}_i^{(l)} \right]^2$$

$$= \frac{1}{2} \sum_{i=1}^{m} \left[ y_i^{(l)} - f\left[ \sum_{j=0}^{\tilde{n}} W_{ij} \cdot f\left[ \sum_{k=0}^{n} w_{jk} \cdot x_k^{(l)} \right] \right] \right]^2 \quad \ldots (8)$$

So, $$w_{jk} = w_{jk} - \eta \frac{\partial J^{(l)}}{\partial w_{jk}}$$

what is $\dfrac{\partial J^{(l)}}{\partial w_{jk}}$

$$\frac{\partial J^{(l)}}{\partial w_{jk}} = \sum_{i=1}^{m} \frac{\partial J^{(l)}}{\partial e_i^{(l)}} \cdot \frac{\partial e_i^{(l)}}{\partial \hat{y}_i^{(l)}} \cdot \frac{\partial \hat{y}_i^{(l)}}{\partial s_i^{(l)}} \cdot \frac{\partial s_i^{(l)}}{\partial h_j^{(l)}} \cdot \frac{\partial h_j^{(l)}}{\partial s_j^{(l)}} \cdot \frac{\partial s_j^{(l)}}{\partial w_{jk}}$$

$$= \sum_{i=1}^{m} e_i^{(l)} \cdot (-1) \cdot f'(s_i^{(l)}) \cdot W_{ij} \cdot f'(s_j^{(l)}) \cdot x_k^{(l)}$$

$$= -\left[ \sum_{i=1}^{m} e_i^{(l)} \cdot f'(s_i^{(l)}) \cdot W_{ij} \right] f'(s_j^{(l)}) \cdot x_k^{(l)}$$

So,

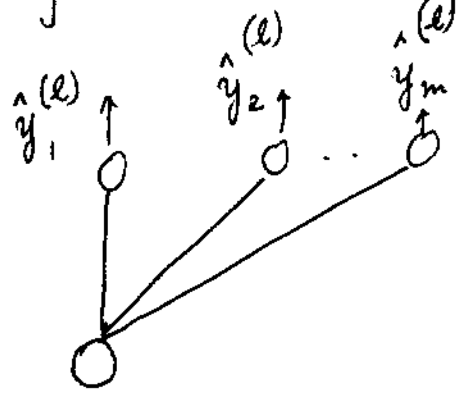
$$w_{jk} = w_{jk} - \eta \frac{\partial J^{(l)}}{\partial w_{jk}}$$

$$= w_{jk} + \eta \left[ \sum_{i=1}^{m} e_i^{(l)} \cdot f'(s_i^{(l)}) \cdot W_{ij} \right] f'(s_j^{(l)}) \cdot x_k^{(l)}$$

$$= w_{jk} + \eta \underbrace{\left[ \sum_{i=1}^{m} \delta_i^{(l)} \cdot W_{ij} \right]}_{\delta_j^{(l)}} f'(s_j^{(l)}) \cdot x_k^{(l)} \quad \cdots (9)$$

---

So, $\delta_j^{(l)}$ is computed as




$\delta_i^{(l)}$ is back-propagated through $W_{ij}$ and summed to give $\delta_j^{(l)}$

The name back-propagation should thus become clear.

## The Complete Algorithm

1) Initialize weights to small random values.
2) for $l = 1, 2, \cdots N$

2a) Do the forward pass.

$$h_j^{(l)} = f\left[S_j^{(l)}\right] = f\left[\sum_{k=0}^{n} w_{jk}\, x_k^{(l)}\right]$$

and,
$$\hat{y}_i^{(l)} = f\left[S_i^{(l)}\right] = f\left[\sum_{j=0}^{\tilde{n}} W_{ij}\, h_j^{(l)}\right]$$

2b) Do the backward pass

$$\delta_i^{(l)} = \left[y_i^{(l)} - \hat{y}_i^{(l)}\right] \cdot f'(S_i^{(l)})$$

and,
$$\delta_j^{(l)} = \sum_{i=1}^{m} \delta_i^{(l)} \cdot W_{ij}$$

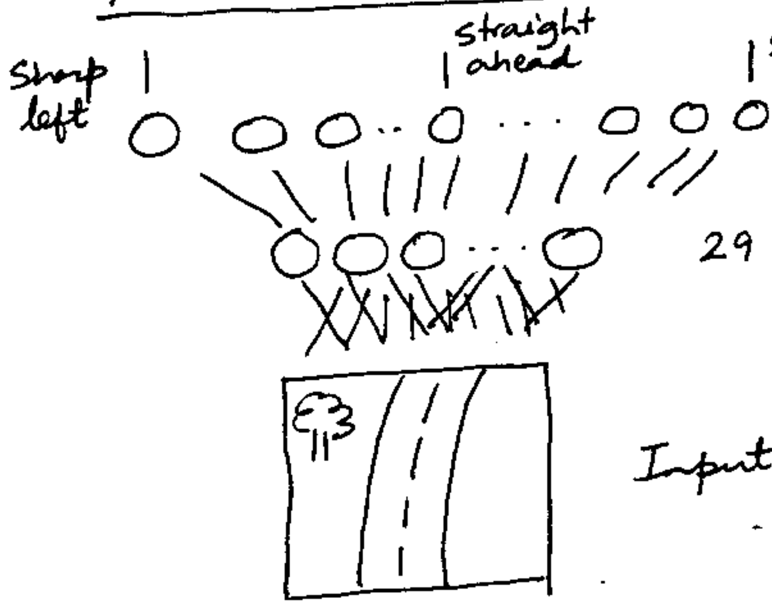
2c) Change weights

$$W_{ij} = W_{ij} + \eta\, \delta_i^{(l)}\, h_j^{(l)}$$
$$w_{jk} = w_{jk} + \eta\, \delta_j^{(\mu)} \cdot x_k^{(l)}$$

3) Present next pattern i.e. goto 2.
4) After an epoch, check $J$. If acceptable stop else set $l=1$ and goto step 2.

# Some Examples of Applications of Feed-Forward Networks

## Autonomous Navigation (Pomerleau)




45 output neurons

29 hidden neurons

Input 30 × 32 pixel image from a video Camera

8 × 32 image from a range finder

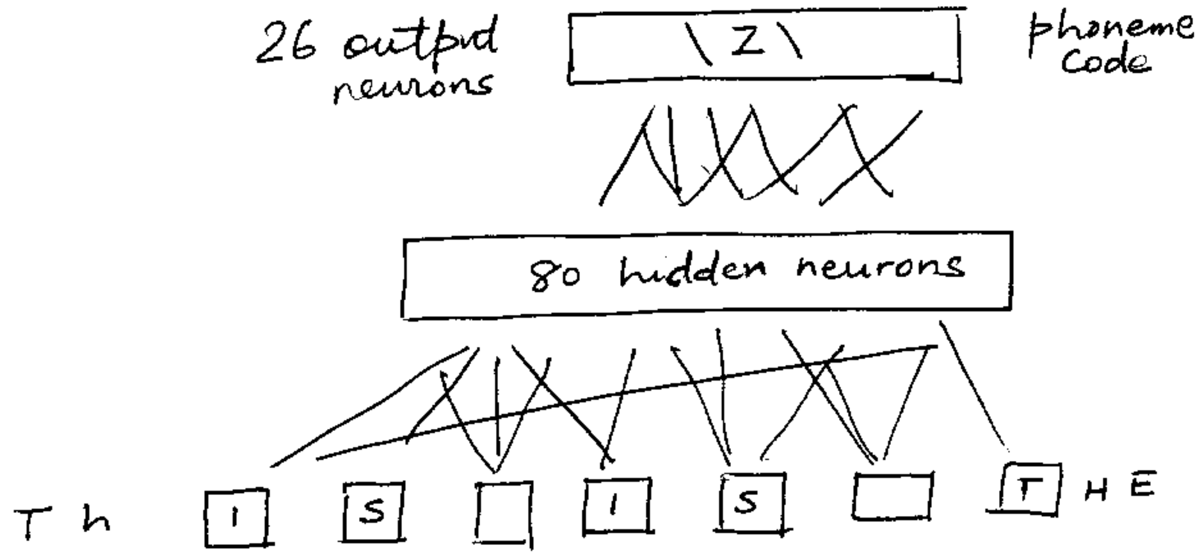
Training 1200 simulated road images

after 40 epochs, the car could autonomously drive at 5 Km/hr (3 mi/hr) on a road through a wooded area

Speed is limited by a SUN-3 computer to do a forward pass

speed was twice as fast as that obtained by any other approach.

# NETtalk (Sejnowski and Rosenberg 1987)




7x29 inputs (seven consecutive Characters presented in a moving fashion)

- Desired output was a phoneme Code for the letter in the Center of the window.

- Network trained using 1024 words. obtained intelligible speech after 10 training epochs (95% accuracy after 50 epochs)

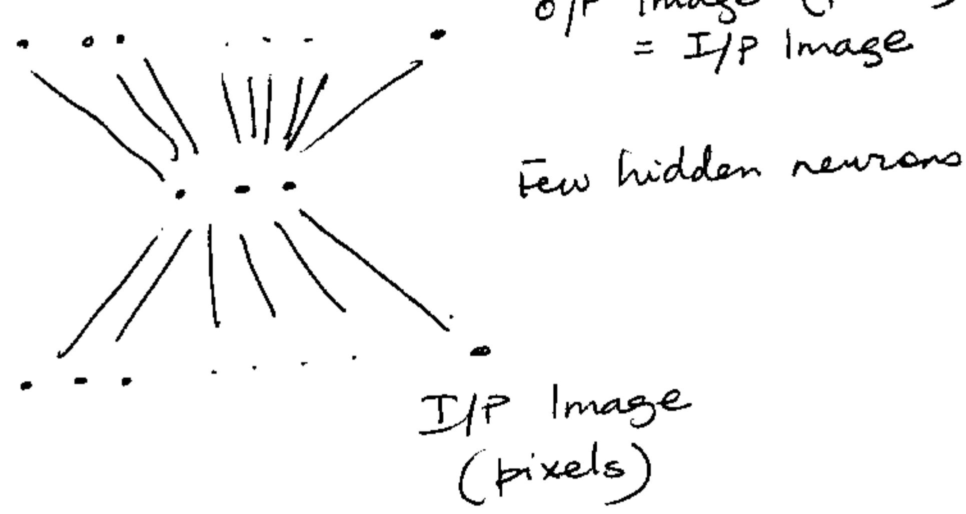
- started off like a child learning to talk and then did better.

- Some hidden neurons were found to represent meaningful properties such as the distinction between Vowels and Consonants.

From HKP

" It is interesting to compare NETtalk with the commercially available DECtalk which is based on hand coded linguistic rules. There is no doubt that DECtalk performs better than NETtalk but this should be seen in the context of the effort involved in creating each system. Whereas NETtalk learned from examples, the rules embodied in DECtalk are the result of about a decade of analysis by many linguists. This exemplifies the ultility of Neural Networks: they are easy to construct and can be used even when a problem is not fully understood. However rule based algorithms usually out perform Neural Networks when enough understanding is available!"

## Image Compression




Compression Phase: Train a network with patches of an image
O/P = I/P.

If the output of the hidden neurons are used, they represent a compressed code of the input (remember there are few hidden neurons)

Decompression use the hidden neuron O/P's and obtain the decompressed image at the O/P layer.

---

Many many other applications in Controls, Geology, Oil exploration, textiles, design, etc etc etc

# Limitations of Back-Propagation

1. Speed of training (slow)
   Once trained outputs are available instantaneously.
2. Temporal Instability (moving target problem)
3. Local minima
4. Good network size (# of hidden layer neurons to use) is not known.

   However one can use pruning, growing, lateral weights for soft weight sharing.

## Advantages of FFNN's

1. General (Universal approximation)
2. Scaling - Scaling - Scaling
3. When regularization is used, provides the best overall generalization performance

# The Response of a Model

Usually, $\mathbf{y}^{(i)} = f(\mathbf{x}^{(i)}) + \epsilon$

Denote the $t$ repeated observations corresponding to this input by $\mathbf{y}^{(1)}, \mathbf{y}^{(2)}, \ldots, \mathbf{y}^{(t)}$. The optimal response $\widehat{f}^*(\mathbf{x}, \Theta)$ should minimize

$$
\begin{aligned}
J \;=\; & \| \widehat{f}^*(\mathbf{x}; \Theta) - \mathbf{y}^{(1)} \|^2 + \| \widehat{f}^*(\mathbf{x}, \Theta) - \mathbf{y}^{(2)} \|^2 + \\
& \ldots + \| \widehat{f}^*(\mathbf{x}, \Theta) - \mathbf{y}^{(t)} \|^2
\end{aligned}
$$

Minimum of $J$ is achieved when $\widehat{f}^*(\mathbf{x}, \Theta) = (\mathbf{y}^{(1)}, \mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(1)})/t$, i.e. $E[\mathbf{y}|\mathbf{x}]$

# The Bias Variance Decomposition

$$
\begin{aligned}
PE &= E_{\mathcal{X}}\left[\| \left(\hat{f}(\mathbf{x},\mathcal{X}) - E_{\mathcal{X}}\left[\hat{f}(\mathbf{x},\mathcal{X})\right]\right) + \left(E_{\mathcal{X}}\left[\hat{f}(\mathbf{x},\mathcal{X})\right] - E[\mathbf{y}|\mathbf{x}]\right) \|^2\right] \\
&= \underbrace{\| E_{\mathcal{X}}\left[\hat{f}(\mathbf{x},\mathcal{X})\right] - E[\mathbf{y}|\mathbf{x}] \|^2}_{\text{Squared Bias}} + \underbrace{E_{\mathcal{X}}\left[\| \hat{f}(\mathbf{x},\mathcal{X}) - E_{\mathcal{X}}\left[\hat{f}(\mathbf{x},\mathcal{X})\right] \|^2\right]}_{\text{Variance}}
\end{aligned}
$$

Large k in k-NN leads to an increase
in variance, small k in large bias. This
Is typical – more complex models have
larger variance and lower bias. The
converse is true for simpler models.