#3) i) Space (storage) complexity:

\# of adjustable weights from input to hidden layer is: $I \cdot J$

\# of adjustable weights from hidden to output layer is: JK

Assuming thresholds are all at zero, total # weights =

$$I \cdot J + J \cdot K$$

During learning, however, one also has to store the gradients $\Longrightarrow$ total storage for weights + their gradients is: $2(IJ + JK)$.

We also need to store the training patterns $\Longrightarrow$

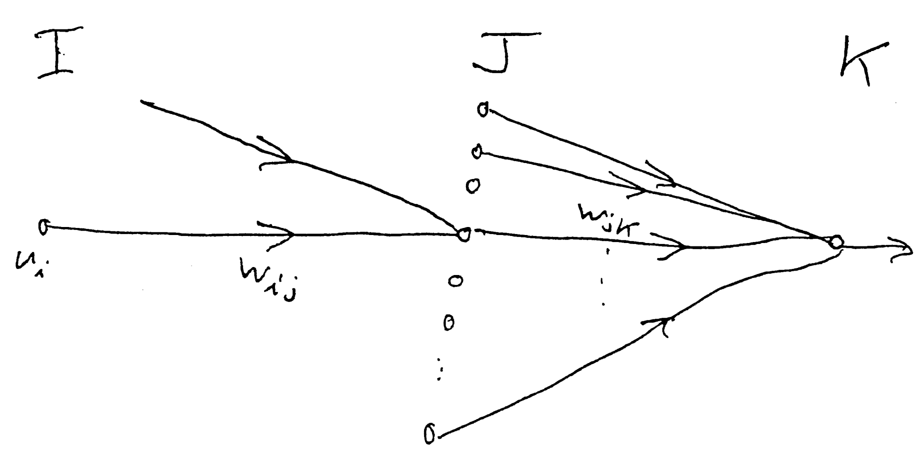
$$(I + 1)T$$

input, output dim., # of training examples

$\Longrightarrow$ space complexity $= 2(IJ + JK) + (I+1)T$

ii) <u>Computational complexity of BP</u>:

Refer to the BP derivation from lecture:



(assume single output, y)

$$v_j = \sum_{i=1}^{I} w_{ij} u_i, \quad j = 1, \dots J \Rightarrow O(IJ) \text{ operations, per sample}$$

$$\underbrace{y = g\left(\sum_{j=1}^{J} w_{jo}\, g(v_j)\right)}_{\text{Forward operations}} \Longrightarrow O(J) \text{ operations, per sample}$$

(mults, & adds).

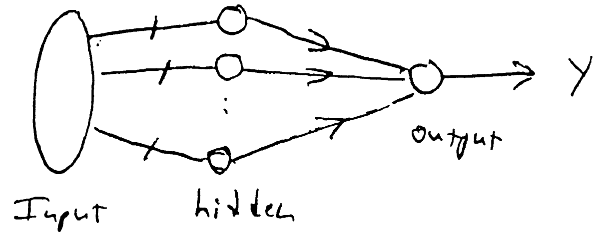
$$\delta_j = g'(v_j)\,\delta_o\, w_{jo}, \quad j = 1, \dots J \Rightarrow O(J) \text{ operations}$$

$$\Delta w_{ij} \propto \delta_j \cdot u_i, \quad i = 1, \dots I, \; j = 1, \dots J \Rightarrow O(IJ) \text{ operations}$$

$$\Delta w_{jo} \propto -\frac{\partial E}{\partial v_o} \cdot \frac{\partial v_o}{\partial w_{jo}}, \quad j = 1, \dots J \Rightarrow O(J) \text{ operations}$$

overall complexity is $O(I \cdot J \cdot T)$ operations

#4) Consider the MLP:



Suppose each neuron uses an <u>antisymmetric</u> activation function, i.e. $f(-x) = -f(x)$ (sigmoid, tanh, or $_{-1}\;{}^{+1}$).

The output can be written

$$y = f\left(\sum_{j=1}^{N_h} \underset{\substack{\downarrow \\ \text{output}}}{w_{jo}}\, f\left(\sum_{i=1}^{d} x_i w_{ij} + w_{j0}\right) + w_0\right)$$

Spse. we flip the sign on all the weights.

Then, neuron j's output becomes

$$f\left(\sum_{i=1}^{d} x_i(-w_{ij}) + (-w_{j0})\right) = -f\left(\sum_{i=1}^{d} x_i w_{ij} + w_{j0}\right)$$

The overall output becomes:

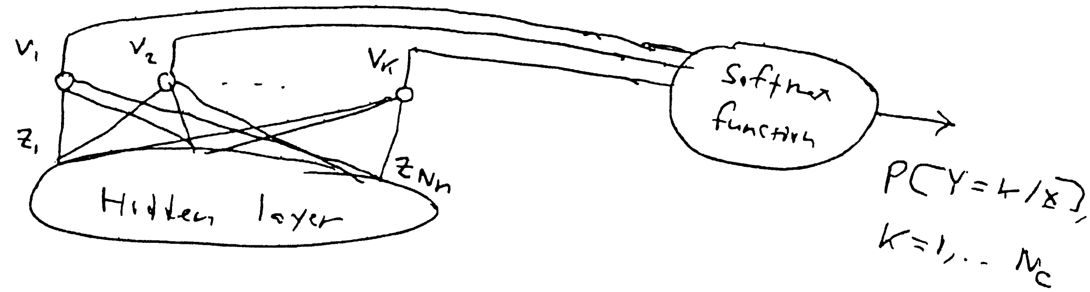
$$f\left(\sum_{j=1}^{N_h} (-w_{jo})\left(-f\left(\sum_{i=1}^{d} x_i w_{ij} + w_{j0}\right)\right) - w_0\right)$$

$$= f\left(\sum_{j=1}^{N_h} w_{jo}\, f\left(\sum_{i=1}^{d} x_i w_{ij} + w_{j0}\right) - w_0\right).$$

The output stays the same <u>if</u> $w_0 = 0$ -- otherwise the operation is in fact changed.

This property does NOT extend to flipping the sign on the input.

#5) Let's just consider the signals from the hidden layer to the softmax function:



Let's denote the outputs of the hidden layer by:

z_j, $j=1,\dots N_h$.

Then, $v_K = \sum_{j=1}^{N_h} z_j w_{jK}$ and $P(Y=k/\underline{x}) = \dfrac{e^{v_K}}{\sum_{K'=1}^{N_c} e^{v_{K'}}}$

As derived in lecture, the cross entropy objective function can be written as:

$$E = -\sum_{c=1}^{N_c} \sum_{\underline{x}:\, c(\underline{x})=c} \log P(Y=c/\underline{x})$$

Now, $\Delta w_{jK} \propto \dfrac{\partial E}{\partial v_K} \cdot \underbrace{\dfrac{\partial v_K}{\partial w_{jK}}}_{z_j}$

$$\frac{\partial E}{\partial v_K} = -\sum_{\underline{x}:\, c(\underline{x})=k} \frac{\partial}{\partial v_K}\left(\log P(Y=k/\underline{x})\right) - \sum_{\substack{c=1,\\ c\neq k}}^{N_c} \sum_{\underline{x}:\, c(\underline{x})=c} \frac{\partial}{\partial v_K}\left(\log P(Y=c/\underline{x})\right),$$

where:

$$\frac{\partial}{\partial v_K}\left(\log P(Y=k/\underline{x})\right) = \frac{\partial}{\partial v_K}\left(v_K - \log\left(\sum_{K'=1}^{N_c} e^{v_{K'}}\right)\right)$$

$$= 1 - \frac{e^{v_K}}{\sum_{K'=1}^{N_c} e^{v_{K'}}} = 1 - P(Y=k/\underline{x})$$

#6) Neural Network Inversion

i) $\underline{X} \longrightarrow \boxed{\text{MLP}} \longrightarrow \underline{Y}$

$\underline{Y} = f(\underline{X})$, with $f(\cdot)$ defined by the MLP. Ideally, if we were given a vector $\underline{Y}$, we could find $\underline{X}$ via $\underline{X} = f^{-1}(\underline{Y})$. Unfortunately,

i) there are/may be many possible $\underline{X}$ that could lead to same $\underline{Y}$ $\Rightarrow$ non-unique inverse;

ii) there may be no $\underline{X}$ leading to a particular $\underline{Y}$.

iii) In general, there is no analytical form for $f^{-1}(\cdot)$.

So, how to find an approximate inverse?

Ans: Use backpropagation!

Let $E = \|\underline{Y} - f(\underline{X})\|^2$ + choose $\underline{X}$ to minimize E.

Algorithm:

Note: strong dependence on initialization...

$t=0$; $\underline{X}^t = \underline{X}_0$ (Init)

loop

$$X_{t+1} = X_t - \mu \nabla_{\underline{X}} E(\underline{X}) \Big|_{\underline{X} = \underline{X}_t}$$

$t \leftarrow t+1$

end loop

Two applications: 1) signal/image reconstruction from noisy, blurred output image,

2) Boundary-finding in classification -- given an MLP classifier, find $\underline{X}$ that cause output to be $\approx 1/2$ (points on the boundary ...).