1 Backpropagation

Problem 1: Apply the basic backpropagation algorithm to the network in Figure 1, with the identity $\sigma(x) = x$ as the activation function on the outputs and $h(x) = tanh(x) = sinh(x)/cosh(x)$ as the activation function of the hidden neurons.

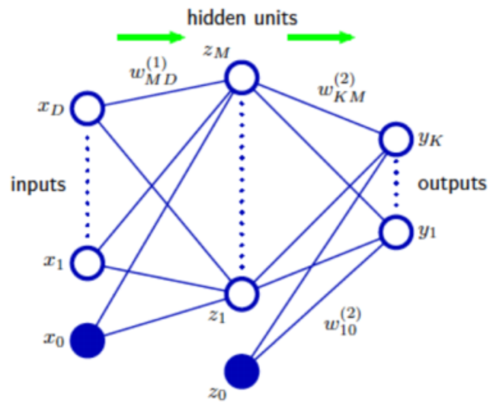
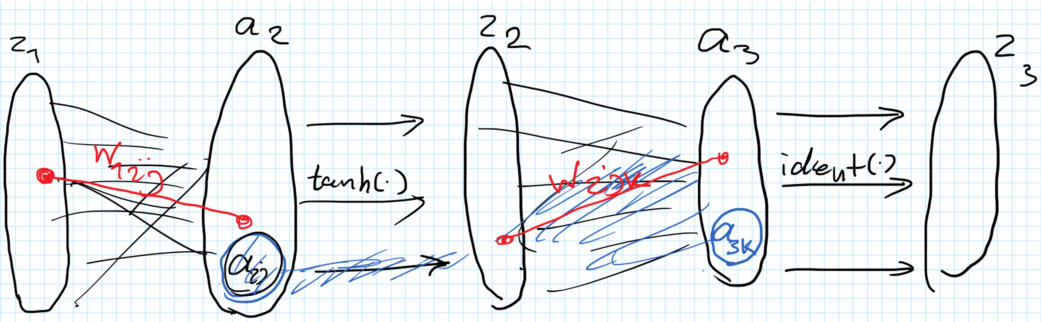


Figure 1: Source Bishop: Figure 5.1

Problem 1- Backpropagation

Monday, 5 November 2018 16:01



$$z_1^{(n)} = x_1^{(n)} \qquad a_2^{(n)} = W_1^T z_1^{(n)} \qquad z_2^{(n)} = \tanh\left(a_2^{(n)}\right)$$

$$a_3^{(n)} = W_2^T z_2^{(n)} \qquad z_3^{(n)} = a_3^{(n)}$$

$$\delta_{3k} = \frac{\partial E_n}{\partial a_{3k}} = \frac{\partial E_n}{\partial z_{3k}} \cancel{\frac{\partial z_{3k}}{\partial a_{3k}}}^{1} = \frac{\partial}{\partial z_{3k}} \left(z_{3k}^{(n)} - t_{nk}\right)^2 \frac{1}{2} = z_{3k}^{(n)} - t_{nk}$$

$$\delta_{2j} = \delta_3^{(n)T} W_{2j:} \tanh'(a_{2j}) = \delta_3^{(n)T} W_{2j:} \left(1 - \tanh(a_{2j})^2\right)$$

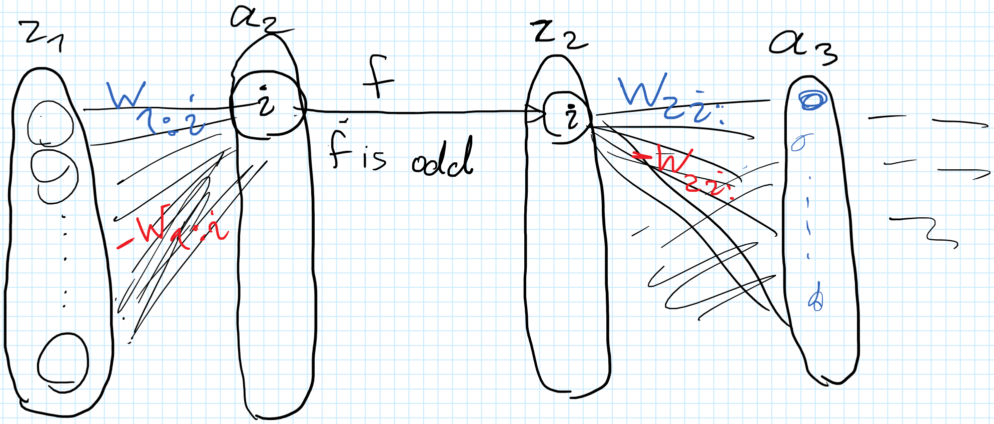
$$\frac{\partial E_n}{\partial w_{ij}} = z_{1i}^{(n)} \delta_{2j}^{(n)} \qquad \frac{\partial E_n}{\partial W_{2jk}} = z_{2j}^{(n)} \delta_{3k}^{(n)}$$

Problem 2- Weight-space Symmetries

Monday, 5 November 2018 16:01

2 Weight Space Symmetries

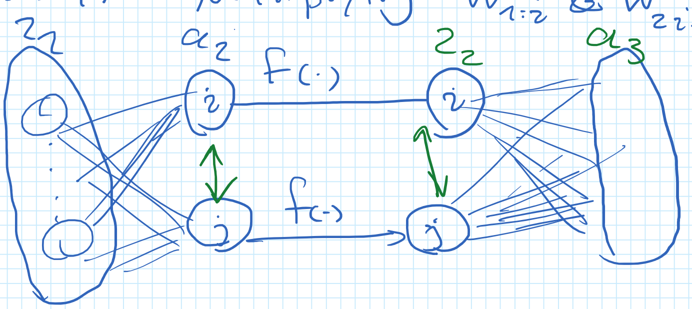
Problem 2: Assume a neural network has odd functions as non-linearities (i.e. functions for which $f(-x) = -f(x)$; e.g. $tanh$). How many equivalent weight sets exist in such a neural network by considering possible changes in signs of weights and permutations among the weights?



$$a_{2i} = W_{1:i}^T z_1 \qquad z_{2i} = f(W_{1:i}^T z_1) \qquad a_{3j} = W_{2ij} f(W_{1:i}^T z_1)$$

$$a_{3j} = -W_{2ij} f(-W_{1:i}^T z_1) = W_{2ij} f(W_{1:i}^T z_1)$$

Operation 1: Muliplying $W_{1:i}$ & $W_{2i:}$ by -1



f has to be odd.

It is the case for e.g., tanh(.).

Operation 2: Swap $[a_{2i}, z_{2i}]$ with $[a_{2j}, z_{2j}]$

(Like the swap shown by the green arrows.)

More precisely, we change the weights W to $\tilde{W}$ as follows:

More precisely, we change the weights W to $\tilde{W}$ as follows:

$$\tilde{W}_{1:i} = W_{1:j}, \quad \tilde{W}_{1:j} = W_{1:i}, \quad \tilde{W}_{2i:} = W_{2j:}, \quad \tilde{W}_{2j:} = W_{2i:}.$$

– About the notation: by $W_{1:j}$, we mean $\begin{bmatrix} W_{11j} \\ W_{12j} \\ \vdots \\ W_{1Dj} \end{bmatrix}$

Assume we have a network like: [linear tanh] [line tanh]
with M hidden neurons.

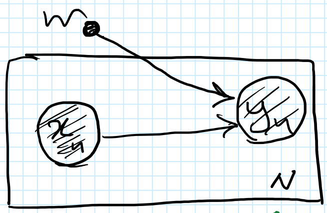
How many equivalen networks can we get?

2^M For operation 1

$M!$ For Operation 2

3 Error functions

Problem 3: Show that maximizing likelihood for a multi-class neural network model in which the network outputs that have the interpretation $f_k(\boldsymbol{x}, \boldsymbol{w}) = p(y_k = 1 \mid \boldsymbol{x})$ are represented by a softmax function is equivalent to the minimization of the cross-entropy error function. We assume that the class labels y of the training dataset $\{x, y\}$ are one-hot-encoded ($y_k \in \{0, 1\}$ and $\sum_{k=0}^{K} = 1$).



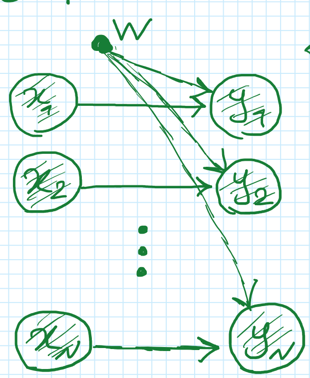
y_n : 1-of-K coding

$y_{nk} = 1 \iff x_n \in \text{Class}_k$

$$y_n \sim \text{Categorical}(y_{n1}, \dots, y_{nK} \mid f_1(x, w), \dots, f_K(x, w))$$

Side note about this figure:

This graphical model is equivalent to this figure:



But using the above form (known as plate notation) is more convenient and more common.

Learning w using MLE:

$$p(\text{All}) = \prod_n p(y_n \mid x_n) \cancel{\prod_n p(x_n)}$$

$$= \prod_n \text{Categorical}(y_{n1}, \dots, y_{nK} \mid f_1(x, w), \dots, f_K(x, w))$$

$$= \prod_n \prod_k f_k(x, w)^{y_{nk}}$$

$$\Rightarrow \text{Log Likelihood} = \sum_n \sum_k y_{nk} \log f_k(x, w) = (-1)\,\text{Cross\_Entropy}$$

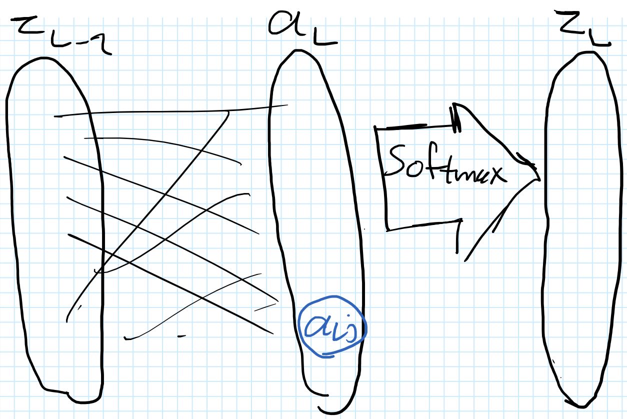
$$w_* = \underset{w}{\operatorname{argmax}}\ \text{Log Likelihood} = \underset{w}{\operatorname{argmin}}\left[\text{Cross\_Entropy}\right]$$

Problem 4: Show that the derivative of the standard (multi-class) cross-entropy error function

$$E(w) = \sum_{n=1}^{N} E_n(w) = -\sum_{n=1}^{N}\sum_{k=1}^{K} y_k^{(n)} \log f_k(\mathbf{x}^{(n)}, \mathbf{w})$$

with respect to the activation a_k for the output units with a softmax activation function satisfies

$$\frac{\partial E_n}{\partial a_k} = f_k(\mathbf{x}^{(n)}, \mathbf{w}) - y_k^{(n)}$$



$$z_{Lk} = \frac{\exp(a_{Lk})}{\sum_j \exp(a_{Lj})}$$

$$\underbrace{\frac{\partial E_n}{\partial a_L}}_{1\times K} = \underbrace{\frac{\partial E_n}{\partial z_L^{(n)}}}_{1\times K} \underbrace{\frac{\partial z_L^{(n)}}{\partial a_L}}_{K\times K}$$

Recall: Chain_rule in higher_dim space.

$A \in \mathbb{R}^I$, $B \in \mathbb{R}^J$, $C \in \mathbb{R}^K$

$$\underbrace{\frac{\partial C}{\partial A}}_{K\times I} = \underbrace{\frac{\partial C}{\partial B}}_{K\times J} \underbrace{\frac{\partial B}{\partial A}}_{J\times I}$$

$$\Rightarrow \frac{\partial E_n}{\partial a_{Lj}} = \frac{\partial E_n}{\partial z_L^{(n)}} \frac{\partial z_L^{(n)}}{\partial a_{Lj}}$$

$$\frac{\partial E_n}{\partial z_{Li}^{(n)}} = \frac{\partial}{\partial z_{Li}^{(n)}}\left[-\sum_{k=1}^{K} y_{nk} \log(z_{Lk}^{(n)})\right] = -\frac{y_{ni}}{z_{Li}^{(n)}}$$

$$\frac{\partial z_{Li}^{(n)}}{\partial a_{Lj}} = \frac{\partial}{\partial a_{Lj}} \frac{\exp(a_{Li})}{\sum_\bullet \exp(a_{L\bullet})}$$

$$i=j \Rightarrow \frac{\partial z_{Li}^{(n)}}{\quad} \quad \exp(a_{Li})\left[\sum_\bullet \exp(a_{L\bullet}) - \exp(a_{Li})\right]$$

$$i = j \Rightarrow \frac{\partial z_{Li}^{(n)}}{\partial a_{Li}} = \frac{\exp(a_{Li})\left[\sum_{\cdot} \exp(a_{L\cdot}) - \exp(a_{Li})\right]}{\left[\sum_{\cdot} \exp(a_{L\cdot})\right]^2}$$

$$= z_{Li} - z_{Li} z_{Li}$$

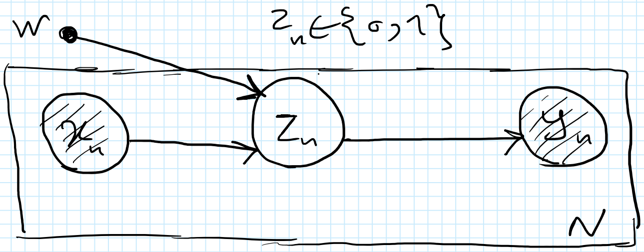
$$i \neq j \Rightarrow \frac{\partial z_{Li}^{(n)}}{\partial a_{Lj}} = -\frac{\exp(a_{Li})\exp(a_{Lj})}{\left[\sum_{\cdot} \exp(a_{L\cdot})\right]^2} = -z_{Li} z_{Lj}$$

$$\Rightarrow \frac{\partial E_n}{\partial a_{Lj}} = \sum_i \left(-\frac{y_{ni}}{\cancel{z_{Li}^{(n)}}}\right)\left(-\cancel{z_{Li}^{(n)}} z_{Lj}^{(n)}\right) - \cancel{z_{Lj}} \frac{y_{nj}}{\cancel{z_{Lj}^{(n)}}}$$

$$= z_{Lj}^{(n)} \left(\sum_i (+ y_{ni})\right)^{\nearrow 1} - y_{nj} = \boxed{z_{Lj}^{(n)} - y_{nj} = \delta_{Lj}^{(n)}}$$

4 Robust classification

Problem 5: Consider a binary classification problem in which the target values are $y \in \{0, 1\}$, with a network output $f(\boldsymbol{x}, \boldsymbol{w})$ that represents $p(y = 1 \mid \boldsymbol{x}, \boldsymbol{w})$, and suppose that there is a probability ε that the class label on a training data point has been incorrectly set. Assuming independent and identically distributed data, write down the error function corresponding to the negative log likelihood. Verify that the well known error function for binary classification is obtained when $\varepsilon = 0$. Note that this error function makes the model robust to incorrectly labelled data, in contrast to the usual error function.



Likelihood is a function of z_n

$$P(z_n \mid x_n) = \text{Bernoulli}(f(x_n, w))$$

$P(y_n \mid z_n) =$

z_n	$P(y_n=0 \mid z_n)$	$P(y_n=1 \mid z_n)$
0	$1-\xi$	ξ
1	ξ	$1-\xi$

To maximize the Likelihood, we need to maximize:

$$\prod_n p(x_n, z_n, y_n)$$

But the problem is that z_n is not observed.

Thus we can maximize: $\prod_n p(x_n, y_n)$

Where $p(x_n, y_n) = p(x_n, z_n = 0, y_n) + p(x_n, z_n = 1, y_n)$

For notational convenience, we write it as:

$$p(x_n, y_n) = p(x_n, z_n = y_n, y_n) + p(x_n, z_n = 1 - y_n, y_n)$$

$\Longrightarrow$

$$P(\text{All}) = \prod_n \left[p(x_n, z_n = y_n, y_n) + p(x_n, z_n = 1 - y_n, y_n) \right]$$

$$= \prod_n \left[p(x_n) \, \text{Bernouli}(y_n, f(x_n, w)) \, (1 - \xi) \right.$$

X

X

Recall:

$\text{Bern}(x; \theta) = \theta^{x} (1-\theta)^{1-x}$

$$+ \, p(x_n) \, \mathrm{Bernoulli}(1-y_n, f(x_n, w)) \, \xi \Big]$$

Recall:

$$\mathrm{Bern}(x;\theta) = \theta^{x}(1-\theta)^{1-x}$$

$$= \prod_n \Big[f(x_n, w)^{y_n} \big(1 - f(x_n, w)\big)^{1-y_n} (1-\xi)$$

$$+ \, f(x_n, w)^{1-y_n} \big(1 - f(x_n, w)\big)^{y_n} (\xi) \Big]$$

In the case $\xi = 0$ $\Rightarrow$ $\mathrm{Cost} = - \prod_n \Big[f(x_n, w)^{y_n} \big(1 - f(x_n, w)\big)^{1-y_n} \Big]$

$$\Rightarrow \mathrm{Log\ Cost} = - \sum_n y_n \log f(x_n, w) + (1-y_n) \log\big(1 - f(x_n, w)\big)$$

which is the binary cross-entropy cost function.