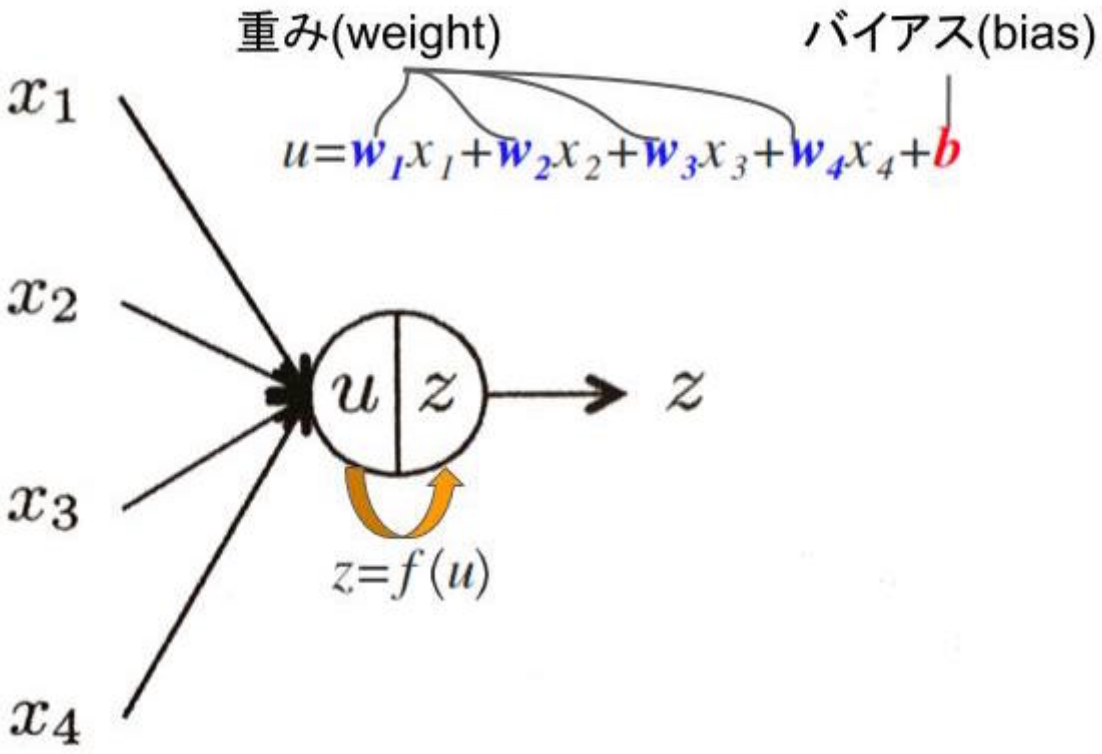
2 順伝播型ネットワーク

順伝播型ネットワークは最も基本的かつ最もよく使われているニューラルネットである。

2.1 ユニットの出力

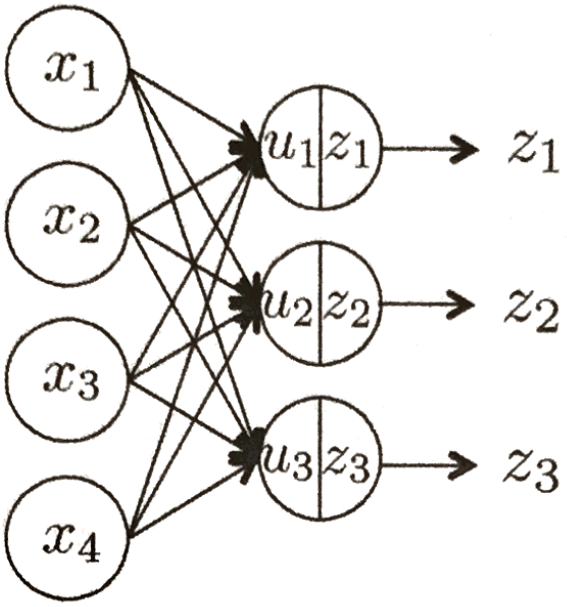
順伝播型(ニューラル)ネットワーク(feedforward neural network): 層状に並べたユニットが隣接層間でのみ結合した構造を持ち、情報が入力層から出力層に一方的にのみ伝播するニューラルネットワーク。または、**多層パーセプトロン**(multi-layer perceptron)とも呼ばれる。



各ユニットは、複数の入力を受け取り、１つの出力を計算する。

- ユニットが受け取る総入力: $u = w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b$
- ユニットの出力: $z = f(u)$
  - $f(u)$: **活性化関数**(activation function)

順伝播型ネットワークでは、各ユニットが層状に並べられ、層間でのみそれらは結合を持つ。



右の層の3つのユニットが受け取る総入力

$$u_1 = w_{11}x_1 + w_{12}x_2 + w_{13}x_3 + w_{14}x_4 + b_1 \qquad (2.1a)$$
$$u_2 = w_{21}x_1 + w_{22}x_2 + w_{23}x_3 + w_{24}x_4 + b_2 \qquad (2.1b)$$
$$u_3 = w_{31}x_1 + w_{32}x_2 + w_{33}x_3 + w_{34}x_4 + b_3 \qquad (2.1c)$$

それぞれの総入力$u_j \quad (j = 1, 2, 3)$に活性化関数を適用したものが出力$z_j \quad (j = 1, 2, 3)$となる。

$$z_j = f(u_j) \quad (j = 1, 2, 3) \qquad (2.2)$$

一般化すると以下のようになる。第1層のユニットを$i = 1, \cdots, I$、第2層のユニットを$j = 1, \cdots, J$で表すと、第1層のユニットの出力から第2層のユニットの出力が決まるまでの計算は

$$u_j = \sum_{i=1}^{I} w_{ji}x_i + b_j$$
$$z_j = f(u_j)$$

行列とベクトルを用いて表記すると

$$\begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_J \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1I} \\ w_{21} & w_{22} & \cdots & w_{22} \\ \vdots & \vdots & \ddots & \vdots \\ w_{J1} & w_{J2} & \cdots & w_{JI} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_I \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_J \end{bmatrix}$$
$$\begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_J \end{bmatrix} = \begin{bmatrix} f(u_1) \\ f(u_2) \\ \vdots \\ f(u_J) \end{bmatrix}$$

2.2 活性化関数

ユニットが持つ活性化関数には通常、単調増加する非線形関数が用いられる。

ロジスティックシグモイド関数(logistic sigmoid function)

あるいは、**ロジスティック関数**

$$f\left(u\right)=\frac{1}{1+e^{-u}}$$

- 定義域: $\left(-\infty,\infty\right)$
- 値域: $\left(0,1\right)$

双曲線正接関数

$$f\left(u\right)=tanh\left(u\right)=\frac{\left(e^{u}-e^{-u}\right)}{\left(e^{u}+e^{-u}\right)}$$

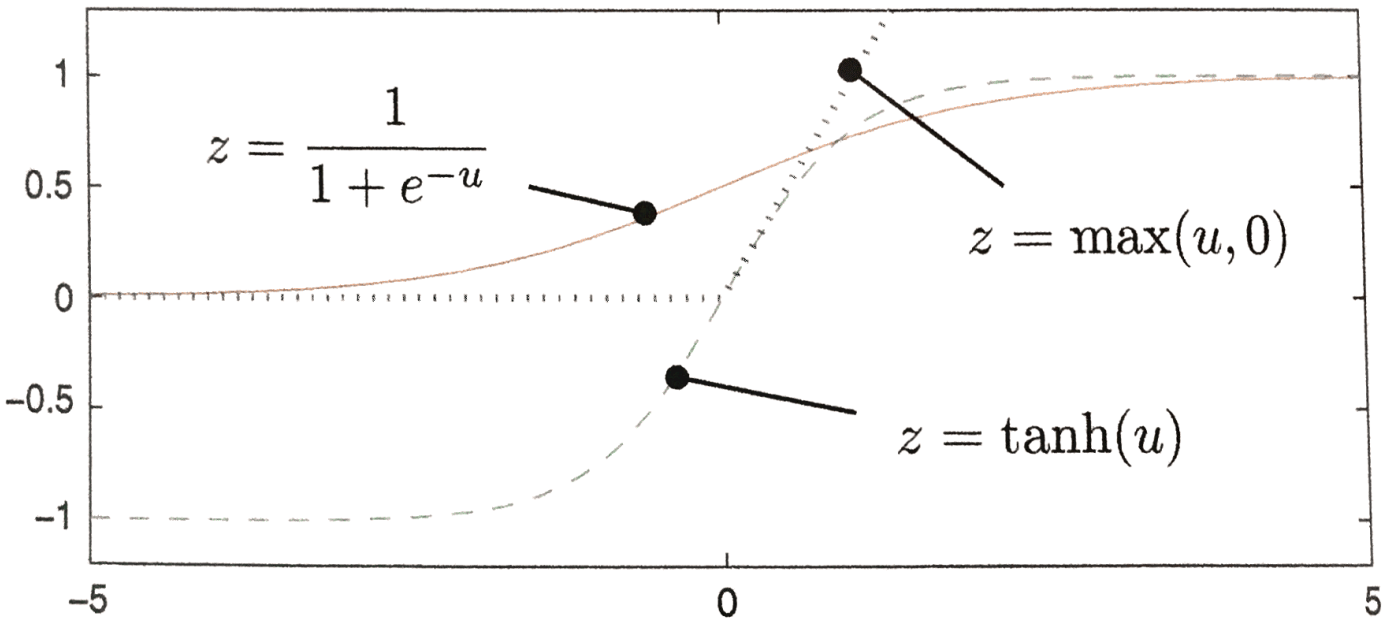
- 値域: $\left(-1,1\right)$

> 上の2つの関数も、
>
> - 入力の絶対値が大きな値をとると出力が飽和し一定値となること
> - その間の入力に対して出力が徐々にかつ滑らかに変化すること
>
> が特徴であり、一般に**シグモイド関数**(sigmoid function)と総称される。

正規化線形関数(rectified linear function)

$$f\left(u\right)=\max\left(u,0\right)$$

- $z=u$の線形関数のうち$u<0$の部分を$u=0$で置き換えただけの単純な関数
- 単純で計算量が小さい
- 上述の2つの関数よりも学習がより速く進み、最終的にもよりよい結果が得られることが多い
- この関数を持つユニットのことを**ReLU**(Rectified Linear Unit)と略記することがある



線形写像・恒等写像

ニューラルネットでは、各ユニットの活性化関数が非線形性を持つことが本質的に重要だが、部分的に線形写像を使う場合がある。

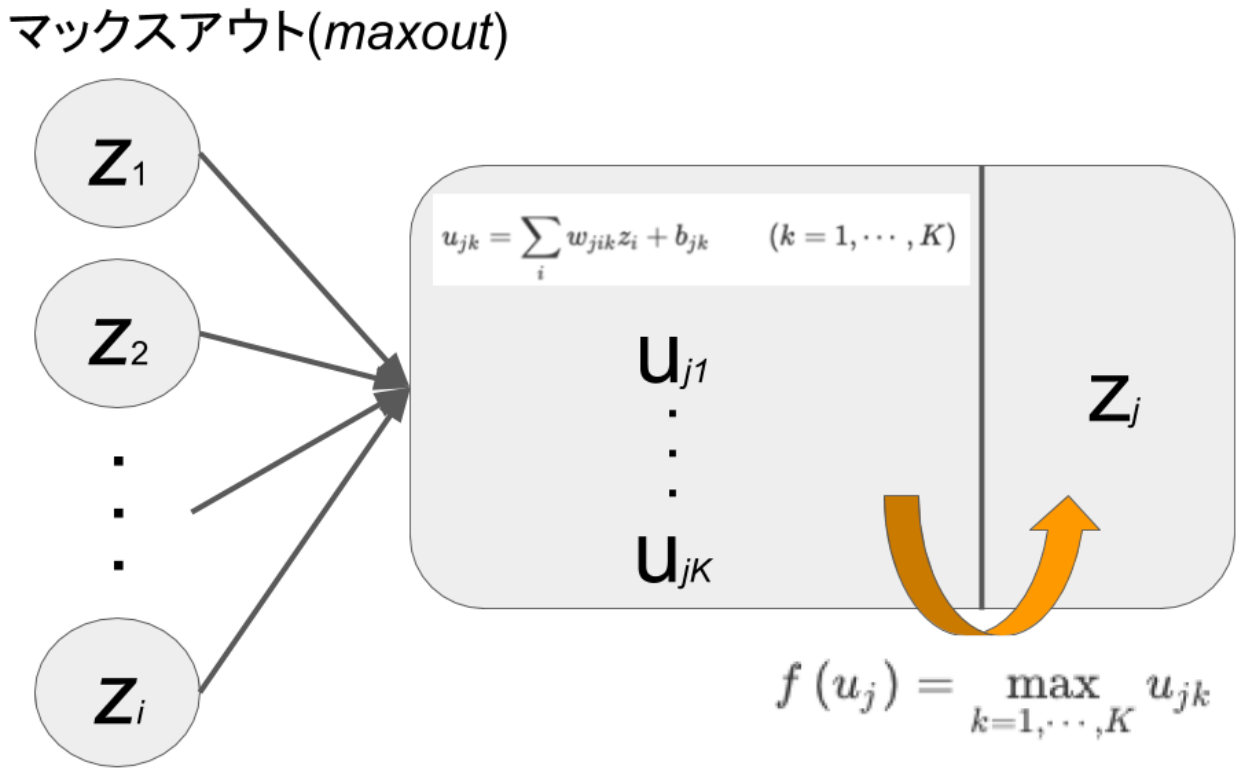
$$f(u) = u \qquad (2.4)$$

- 回帰問題のためのネットワークでは、出力層に恒等写像を用いる

ロジスティック関数を区分的に直線で近似した関数

$$f(u) = \begin{cases} -1 & u < -1 \\ u & -1 \leq u < 1 \\ 1 & u \geq 1 \end{cases}$$

マックスアウト(maxout)



この活性化関数を持つユニット1つは、K個の異なるユニットをまとめて1つにしたような構造を持つ。それらK個の1つ1つが異なる重みとバイアスを持ち、それぞれの総入力を$u_{j1}, \cdots, u_{jK}$と別々に計算した後、それらの最大値をこのユニットの出力とする。

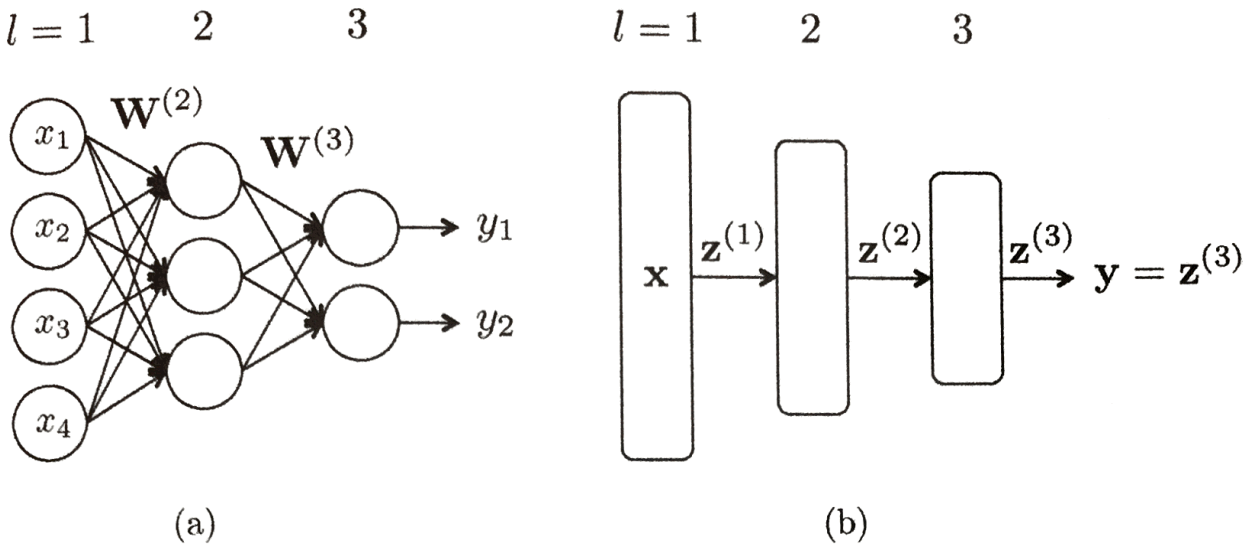
$$u_{jk} = \sum_i w_{jik} z_i + b_{jk} \qquad (k = 1, \cdots, K)$$

$$f(u_j) = \max_{k=1,\cdots,K} u_{jk}$$

各ユニットがこの活性化関数を持つネットワークは、各種ベンチマークテストで、正規化線形関数を使ったネットワークを凌ぐ結果を残している。

2.3 多層ネットワーク

ニューラルネットでは、情報は以下の図のように左から右へと一方向に伝わり、この順に各層を$l = 1, 2, 3$で表す。



なお、$l=1$の層を**入力層**(**input layer**)、$l=2$を**中間層**(**internal layer**)あるいは**隠れ層**(**hidden layer**)、$l=3$を**出力層**(**output layer**)と呼ぶ。

- 各層のユニットの入出力を区別するために、各変数の右肩に層の番号($l=1,2,3$)を付け、$\mathrm{u}^{(l)}$や$\mathrm{z}^{(l)}$のように書くことにする

以下では一般化して、任意の層数Lのネットワークの入出力を表記する。層$l+1$のユニットの出力$\mathrm{z}^{(l+1)}$は、1つ下の層lのユニットの出力$\mathrm{z}^{(l)}$から

$$\mathrm{u}^{(l+1)} = \mathrm{W}^{(l+1)}\mathrm{z}^{(l)} + \mathrm{b}^{(l+1)} \qquad (2.5a)$$
$$\mathrm{z}^{(l+1)} = f\left(\mathrm{u}^{(l+1)}\right) \qquad (2.5b)$$

のように計算される。したがって、ネットワークに対する入力xが与えられたとき(層$l=1$のユニットの出力を$\mathrm{z}^{(1)}=\mathrm{x}$とし)、式(2.5a)、式(2.5b)を$l=1,2,3,\ldots,L-1$の順に実行していくことで、各層の出力$\mathrm{z}^{(2)},\mathrm{z}^{(3)},\ldots,\mathrm{z}^{(L)}$をこの順に決定していくことができます。

一般化された多層ニューラルネットの行列表記

$$\begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_J \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1p} \\ w_{21} & w_{22} & \cdots & w_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ w_{J1} & w_{J2} & \cdots & w_{Jp} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_J \end{bmatrix}$$

(J×1) (J×p) (p×1) (J×1)

以下では、ネットワークの最終的な出力を

$$\mathrm{y} \equiv \mathrm{z}^{(L)}$$

と表記することにする。**なお上では、表記の簡素化のため単一の活性化関数fを使うように表記したが、各層で異なるものを選んでも構わない。特に出力層のユニットの活性化関数は、問題に応じて一般に中間層とは異なるものを選ぶ。**

> **まとめ**
>
> このように順伝播型ネットワークでは、与えられた入力xに対し、入力層側から出力層側へ、上の計算を繰り返すことで情報を伝搬させ、出力yを計算する。

> - この関係は、$\mathrm{y} = f(\mathrm{x})$として表現できる
> - この関数のパラメータは$\mathrm{W}^{(l)}$と$\mathrm{b}^{(l)}$である
> - 活性化関数のパラメータを$f(\mathrm{x}; \mathrm{W}, \mathrm{b})$と表す

2.4 出力層の設計と誤差関数

2.4.1 学習の枠組み

あるxが与えられたときにdを予測するモデルを学習することを目的とする。

> N 個の訓練データを$D = \{(x_i, y_i)\}_{i=1}^{N}$ としたとき、学習データでの誤差(以下の目的関数)を最小化するモデルパラメータ**θ**の値を求める手続きを教師あり学習と呼ぶ。
>
> $$L(\theta) = \frac{1}{|D|} \sum_{i=1}^{|D|} \ell_\theta \left(x^{(i)}, y^{(i)} \right)$$

> $\ell_\theta(x, y) \geq 0$: 個々の事例データに対して定義する**誤差関数**(error function)
>
> ※機械学習ではむしろ損失関数(loss function)が一般的な名称だが、本レポートでは他の名称とも整合性を考えてこのように呼ぶ

以下の表は、主にニューラルネットで用いられる誤差関数である。

問題の種別	出力層の活性化関数	誤差関数
回帰	恒等写像 $f(u) = u$	二乗誤差　式(2.6)

問題の種別	出力層の活性化関数	誤差関数
二値分類	ロジスティック関数 $f\left(u\right)=\frac{1}{1+e^{-u}}$	式(2.8)
多クラス分類	ソフトマックス関数	交差エントロピー 式(2.11)

2.4.2 回帰(regression)

ニューラルネットを用いた回帰(regression)では、他の機械学習アルゴリズムと同じで連続値を出力する。ただし、ネットワークの出力層の活性化関数に、その値域が目標とする関数のそれと一致するようなものを選ぶ必要がある。

値域	活性化関数
$\left[-1:1\right]$	双曲線正接関数
$\left(-\infty,\infty\right)$	恒等写像
$\cdots$	$\cdots$

回帰モデルの学習によく使われるニューラルネットのアルゴリズムは以下のような誤差関数がある。しかし、実際には学習データの出力yが持つ統計的な性質に応じて選ぶのが理想である。yがその真の値にガウス分布に従うノイズが加算されたものと考えられる場合、以下の最小二乗誤差を選ぶのが最適である。

> **最小二乗法**
>
> $$E\left(w\right)=\frac{1}{2}\sum_{i=1}^{N}\left[y_i-f\left(x_i;w\right)\right]^2$$
>
> (1/2倍するのは、微分すると2乗の2と相殺されるからである。)

機械学習で抑えておくべき損失関数（回帰編） - HELLO CYBERNETICS
http://s0sem0y.hatenablog.com/entry/2017/06/19/084210

2.4.3 二値分類

入力xを内容に応じて2種類に分類する問題を考える。この問題を定式化する方法はいくつかあるが、ここではxを指定したとき、$y=1$となる事後確率 $p\left(y=1|x\right)$ をモデル化する方法を考える。与えられたxに対するyの推定は、このモデルを使って事後確率を計算し、

$$y = \begin{cases} 1 & p\left(y=1|x\right) \geq 0.5 \\ 0 & \text{else} \end{cases}$$

と判断することとする。

この事後確率$p\left(y=1|\mathrm{x}\right)$をモデル化するのにニューラルネットを使う。このニューラルネットは、出力層にユニットを1つだけ持ち、その活性化関数はロジスティック関数$y=1/\left(1+\exp\left(-u\right)\right)$とする。このネットワーク全体の入出力関係$f\left(\mathrm{x};w\right)$を事後確率のモデル

$$p\left(y=1|\mathrm{x}\right) \approx f\left(\mathrm{x};w\right)$$

とする。

パラメータwは、訓練データ$\left\{\left(\mathrm{x}_i, y_i\right)|i=1,\cdots,N\right\}$を用いて、モデルが与える事後分布$p\left(y|\mathrm{x};w\right)$が、データが与える分布と最もよく整合するように決める。具体的には、**最尤推定(maximum likelihood estimation)**を行うことを考える。最尤推定は、このモデルの下でwのデータに対する**尤度(likelihood)**を求め、それを最大化するようなwを選ぶ。

$$L\left(w\right)=\prod_{i=1}^{N}p\left(y_i|\mathrm{x}_i;w\right)=\prod_{i=1}^{N}\left\{f\left(\mathrm{x}_i;w\right)\right\}^{y_i}\left\{1-f\left(\mathrm{x}_i;w\right)\right\}^{1-y_i}$$

上記の尤度関数の対数をとり、さらに(最大化の代わりに最小化を考えて)符号を反転した

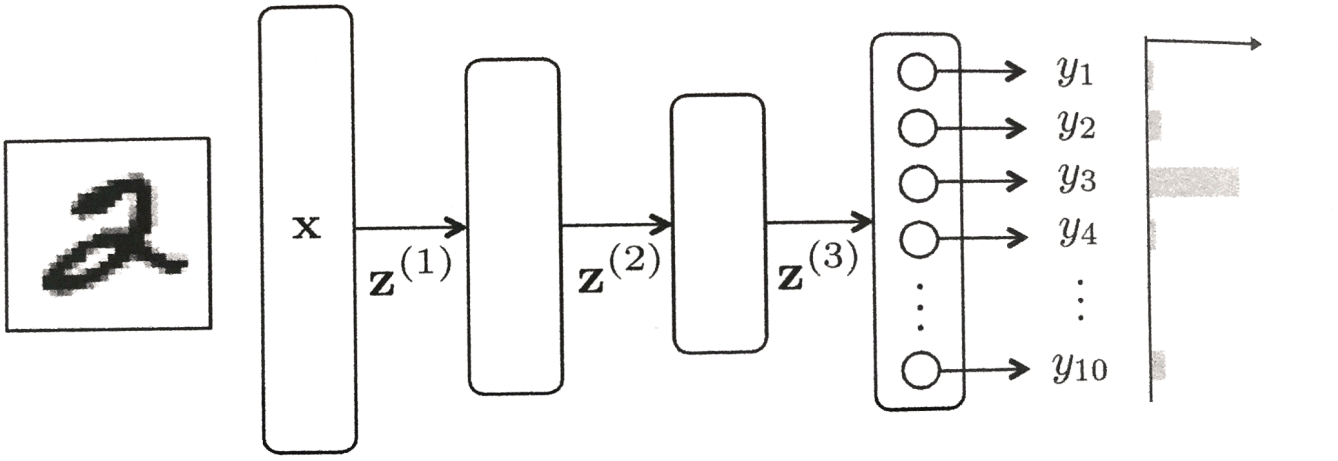
$$E\left(w\right)=-\sum_{i=1}^{N}\left\{y_i\log f\left(\mathrm{x}_i;w\right)+\left(1-y_i\right)\log\left[1-f\left(\mathrm{x}_i;w\right)\right]\right\}$$

を損失関数とする。

機械学習で抑えておくべき損失関数（分類編） - HELLO CYBERNETICS
https://www.hellocybernetics.tech/entry/2017/06/20/135402

2.4.4 多クラス分類

クラス分類とは、入力xを内容に応じて有限個のクラスに分類する問題である。多クラス分類を対象とする場合、ネットワークの出力層に分類したいクラス数Kと同数のユニットを並べ、この層の活性化関数を次のように並べる。



(多クラス問題の例。手書き数字の認識。入力画像xが、0~9までの数字のどれかを判定する。)

出力層$l = L$の各ユニット$k(= 1, \ldots, K)$の総入力は、1つ下の層$l = L - 1$の出力を元に$u_k^{(L)} = \mathrm{W}^{(L)}\mathrm{z}^{(L-1)}\mathrm{b}^{(L)}$と与えられる。これを元に、出力層の$k$番目のユニットの出力を

$$y_k \equiv z_k^{(L)} = \frac{\exp\left(u_k^{(L)}\right)}{\sum_{j=1}^{K}\exp\left(u_j^{(L)}\right)}$$

とする。この関数は**ソフトマックス関数(softmax function)**と呼ばれている。

- 出力$y_1, \ldots, y_K$は、総和がいつも1になる($\sum_{k=1}^{K} y_k = 1$)ことに注意
- ソフトマックス関数も活性化関数の1つに数えられるが、他の活性化関数の場合、ユニットkの出力z_kは同ユニットへの総入力u_kのみから決定されるのに対し、ソフトマックス関数では、ユニットkの出力はこの層の全ユニットへの総入力$u_1, \ldots, u_K$を元に決まる点で特殊である

今考えているクラスを$C_1, \ldots, C_K$と表すとき、上のように選んだ出力層のユニットkの出力$y_k(= z_k^{(L)})$は、与えられた入力xがクラスC_kに属する確率を表すものと解釈する。

$$p\left(C_k|\mathrm{x}\right) = y_k = z_k^{(L)} \qquad (2.10)$$

そして、入力xをこの確率が最大になるクラスに分類することにする。

多クラス分類でも二値分類同様に、ネットワークが実現する関数が各クラスの事後確率のモデルであると見なし、そのような確率モデルの下で、訓練データに対するネットワークパラメータの尤度を評価し、これを最大化する。

訓練データ：

$$y_i = \begin{pmatrix} y_{i1} & y_{i2} & \cdots & y_{iK-1} & y_{iK} \end{pmatrix}^T$$

$$y_i = \begin{pmatrix} 0 & 0 & \dots & 1 & 0 \end{pmatrix}^T$$

事後分布を以下のように表す。

$$p\left(y|\mathrm{x}\right) = \prod_{k=1}^{K} p\left(C_k|\mathrm{x}\right)^{y_k}$$

この事後確率の表記に式(2.10)を組み合わせると、訓練データ$\left\{\left(\mathrm{x}_i, y_i\right)\right\} \quad (i = 1, \dots, N)$に対する$w$の尤度を以下のように導出できる。

$$L\left(w\right) = \prod_{i=1}^{N} p\left(y_i|x_i; w\right) = \prod_{i=1}^{N}\prod_{k=1}^{K} p\left(C_k|x_i\right)^{y_{ik}} = \prod_{i=1}^{N}\prod_{k=1}^{K} \left(y_k\left(x; w\right)\right)^{y_{ik}}$$

この尤度の対数をとり符号を反転した次を、損失関数とする。

$$E\left(w\right) = -\sum_{i=1}^{N}\sum_{k=1}^{K} y_{ik} \log y_k\left(x_i; w\right)$$

この関数は、**交差エントロピー(cross entropy)**と呼ばれる。