第3章

chainerの使い方

14T4071T

渡部勇樹

chainer

- chainerはDeepLearningのフレームワーク
- NNのような演算を行うネットワークで表現される関数のパラメータを、訓練データから学習するプログラムを構築するためのpythonのパッケージ

合成関数

・$z = f(x_1, x_2, x_3) = (x_1 - 2x_2 - 1)^2 + (x_2 x_3 - 1)^2 + 1$

ここで

$h_1(y_1, y_2) = {y_1}^2 + {y_2}^2 + 1$という関数を考えると

$z = f(x_1, x_2, x_3) = h_1(x_1 - 2x_2 - 1, x_2 x_3 - 1)$となる

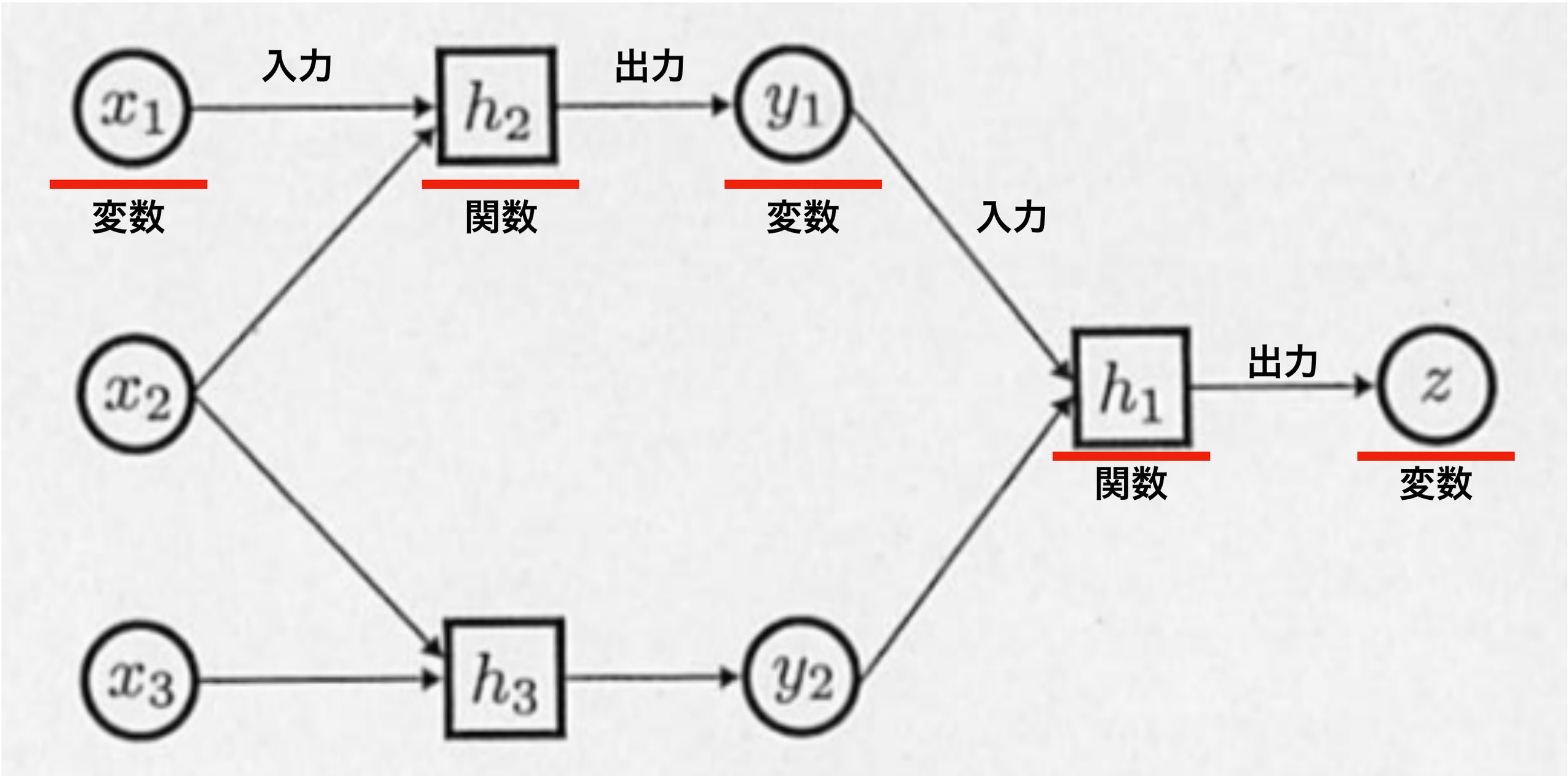
さらに　$h_2(z_1, z_2) = z_1 - 2z_2 - 1,$と定義すると

$h_3(u_1, u_2) = u_1 u_2 - 1$

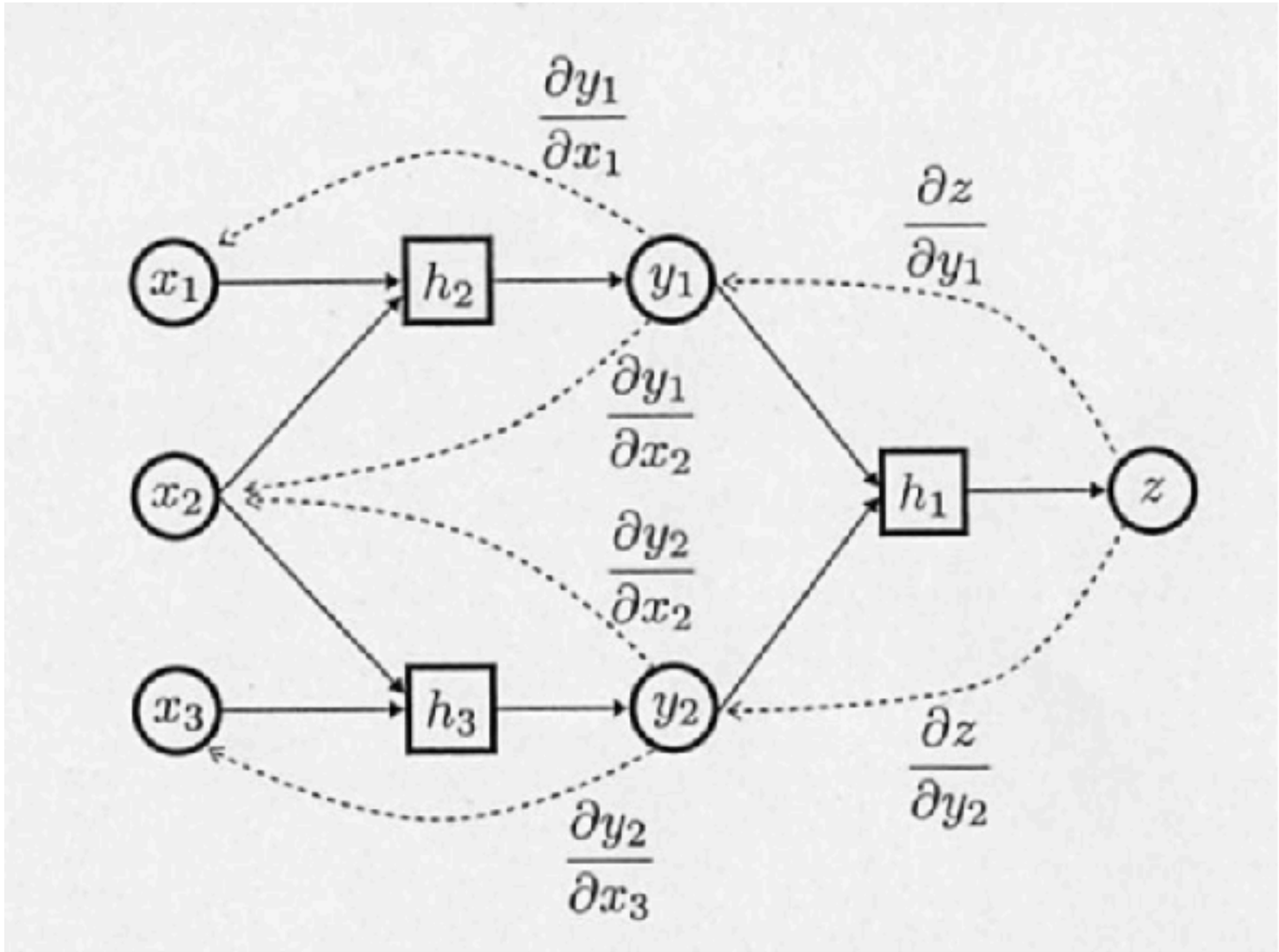
$z = f(x_1, x_2, x_3) = h_1(x_1 - 2x_2 - 1, x_2 x_3 - 1) = h_1(h_2(x_1, x_2), h_3(x_2, x_3))$

計算グラフ

$$z = f(x_1, x_2, x_3) = h_1(x_1 - 2x_2 - 1, x_2 x_3 - 1) = h_1(h_2(x_1, x_2), h_3(x_2, x_3))$$



微分を含んだ計算グラフ



$$\frac{\partial z}{\partial x_1} = \frac{\partial z}{\partial y_1}\frac{\partial y_1}{\partial x_1} \qquad \frac{\partial z}{\partial x_2} = \frac{\partial z}{\partial y_1}\frac{\partial y_1}{\partial x_2} + \frac{\partial z}{\partial y_2}\frac{\partial y_2}{\partial x_2}$$

基本オブジェクト

```
import numpy as np
import chainer
from chainer import cuda, Function, report, training, utils, Variable
from chainer import datasets, iterators, optimizers, serializers
from chainer import Link, Chain, ChainList
import chainer.functions as F
import chainer.links as L
from chainer.training import extensions
```

Variable

- 変数のノードに対応するオブジェクトを生成する
- 実数のタイプは、np.float32
  整数のタイプは、np.int32　　に固定する必要がある
- 型を変換するastypeの方が応用が効く
- Variableの変数の演算結果もまたVariableの変数になる
- 変数の中身はdataという属性で参照できる

Variable

```
x1 = Variable(np.array([1],dtype=np.float32))
x2 = Variable(np.array([2],dtype=np.float32))
x3 = Variable(np.array([3],dtype=np.float32))
```

```
x1 = Variable(np.array([1]).astype(np.float32))#変数の生成
x2 = Variable(np.array([2]).astype(np.float32))
x3 = Variable(np.array([3]).astype(np.float32))

z=(x1-2*x2-1)**2+(x2*x3-1)**2+1#順方向の計算

print 'z',z.data#順方向の計算結果

z.backward()#逆向きの計算を行う（微分値を得る）
print 'backx1',x1.grad#微分値の表示
print 'backx2',x2.grad
print 'backx3',x3.grad
```

functions

- Variableを変数として持つ関数を提供する
- 三角関数などの通常の関数のほか、活性化関数や損失関数など様々な関数が提供されている
- 例）F.sin(x).data

functions

```
x = Variable(np.array([-1,0,1], dtype=np.float32))#変数の生成
z = F.sin(x)#sin関数
z.grad = np.ones(3, dtype=np.float32)#関数の傾きの次元を渡す

z.backward()#逆向きの計算
print x.grad#微分値の表示
```

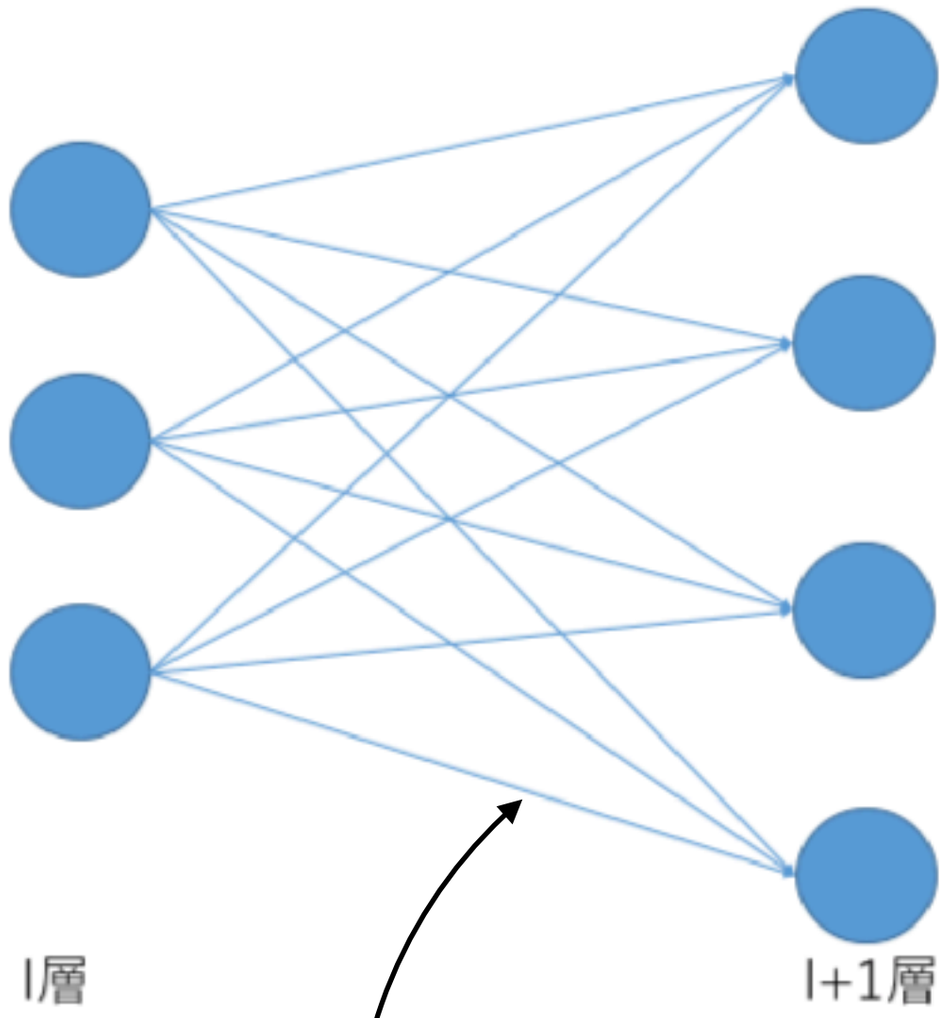
links

- Variableを変数として持つ関数を提供する
- functionsとの違い

  functions：関数にパラメータを持たない

  links　　：関数にパラメータを持つ

links



y = Wx+b　(W,bはパラメータ)

links

```
h = L.Linear(3,4)#linksの定義
x = Variable(np.array(range(6)).astype(np.float32).reshape(2,3))
y = h(x)#代入

print y.data
```

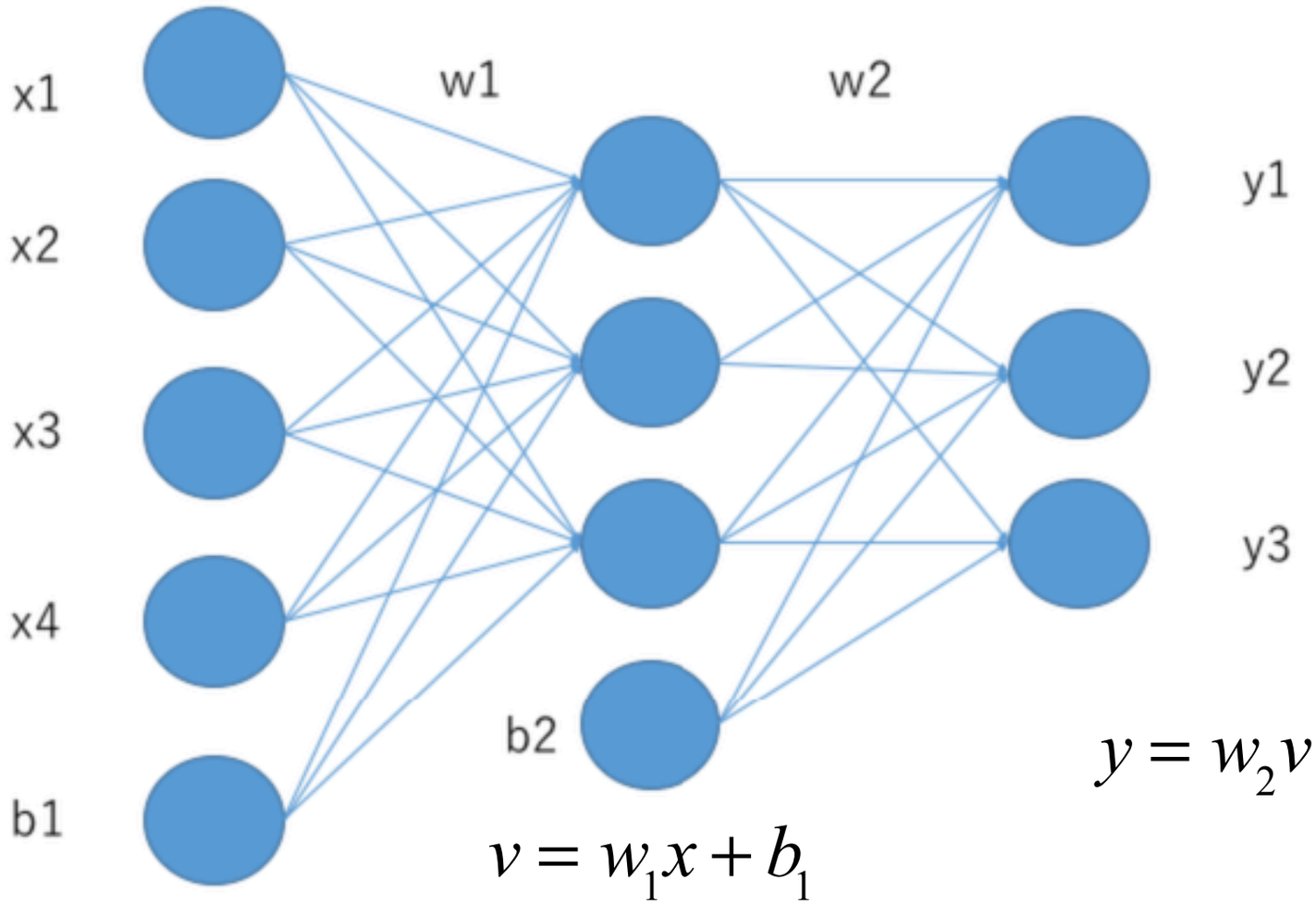
Chain

- ChainではLinks内の関数やFunctions内の関数を合成し、
  その合成関数内に含まれるパラメータを学習

- この合成関数自体が、モデルを表現している

- Chainクラスはモデルを定義するためのクラス

Chain



$$y = w_2(w_1 x + b_1) + b_2$$

$$y = w_2 \delta(w_1 x + b_1) + b_2$$

optimizers

- パラメータを更新して最適化する