

メディア情報学実験 情報検索，認識（画像）

2024/01/25

庄野 逸 (shouno@uec.ac.jp)

スタッフ

- 柳井 啓司 教授（後半 2 回分担当）

yanai@cs.uec.ac.jp

- 庄野 逸（前半 1 回分担当）

shouno@uec.ac.jp

- TA紹介

- 泉 幸太（柳井研 M1）

- 山口 廉斗（柳井研 M1）

- 住谷裕太（庄野研 M2）

- 辻駿哉（庄野研 M2）

実験概要

- 2限は実験概要の説明，3限は各自実験
 - 実験時間以外にも，通信環境を整えて IED を使うことはOK.
- 場所は IED でオンサイトで行なう.
 - 特に席ぎめなどを行わない

実験概要（続き）

- 計算機実験は IED の計算リソース(GPU)を使います.
(自前で計算リソースがあれば, それでもOK)
- Firefox などの Web ブラウザ(Linux 準拠)
- GPU サーバー(IED 側で準備済み)
- GPU サーバーがアクセスできるファイルシステム
(IED 側で準備済み, ファイルの置き場所は後述)
- 演習を提供する jupyter notebook を解釈できる環境
(IED 側で準備済み)

学外からの実験環境: VPN設定について

- ❑ IED の GPU サーバは、学外からは見えないので、学外から実験する際にはVPNを使って学内ネットワークの傘下に入る必要があります。
- ❑ VPNに関しては情報基盤センターの指針に従ってください。
VPNの設定は下記参照
<https://www.cc.uec.ac.jp/ug/ja/remote/vpn/>
- ❑ VPNの傘下に入った場合、proxy を設定しないと学外が見えません(zoom などがアウトに)
- ❑ しかし、proxy を使って大学内部のホストを見ると変なことが起こります
→ proxy の除外設定を適切に
proxy.uec.ac.jp:8080; 172.21.0.0/16;
*.uec.ac.jp

解説文書置き場

- 実験自体は Jupyter Notebook のテンプレートを配りますので、それを使って
 - Python のコードを組み、
 - 機械学習や画像認識の実験に取り組む
 - 実験結果、考察をファイルとしてをアップロードという流れになります。
- 下記に説明などをおいておきます。
<https://uecmediaexp.github.io/IntroductionDocument/>

GPUマシンの確認

- IEDの情報リソースは下記を参照（学内/VPN経由で見れます）

<http://gpu1.ied.inf.uec.ac.jp/>

- {gpu1, gpu2, gpu3, gpu4}.ied.inf.uec.ac.jp (は RTX3080 GPU8個)
- {gpu5, gpu6, gpu7, gpu8}.ied.inf.uec.ac.jp (は A4000 GPU10個)

GPUマシンの割当

□ ゆるい割り当てとして下記の原則を指定しておきます

- 1組 gpu1, gpu5 の18個
学籍番号 211004~2110174 までの19名
- 2組 gpu2, gpu6 の18個
学籍番号 2110185~2110379 までの19名
- 3組 gpu3, gpu7 の18個
学籍番号 2110391~2110605 までの19名
- 4組 gpu4, gpu8 の18個
学籍番号 2110610~2110737, 過年次生3名 (17名) + 教員, TA

GPUマシンの使用時のマナー

- 空いているGPU の Jupyter Hub を用いてください
リソース管理のページで確認できます
- 他の人とバッティングした場合、メモリ不足（Out of Memory）と怒られるケースがあります.
- 計算を終えた場合、あるいは動かさない場合は Jupyter Notebook の “Running” タブから
要らないものを Terminate してください.
- 長時間占有している計算は殺される可能性があります.

やらしてもらおうこと（パターン認識）

- 2024/01/25
回帰・パターン認識問題, ニューラルネットワーク
- 2024/02/01
深層学習 1 (CNN他)
- 2024/02/08
深層学習 2 (発展課題)

提出課題

- 各教員の指示に従ってください

- 提出物

Jupyter 上で作成した ノートブック(.ipynb拡張子) を
PDF に変換したファイルを提出してください.

- 提出方法

- 提出場所は <https://mm.cs.uec.ac.jp/media/>

ここから課題 1 の説明

課題 1 (2023/01/26出題)

- 多層パーセプトロンを構成し,
MNISTデータセットの識別性能を調査する.
- 与えられた Jupyter notebook を用いて答えること
- ニューラルネットワークを触ったことがあるなら一度は通る
途です. わかったヒトはさっくり進めてもらって構いません.
- 演習課題も解いておくこと (解答例ついています).
解いた (写経して) ものも提出課題に含みます.
- 提出期限は 2週間後 (2024/02/08) とします.

課題 1 をみて途方にくれた場合

- 課題 1 を見て途方にくれたヒトもいるかと思います.
- でも, 諦めないでください.
解けるように解答例つき演習問題を設定しています.
- 演習問題1-1 ~ 1-5 を順に解くことは,
課題を解くためのヒントです.
- なお演習問題の例解は PracticeHint ディレクトリにあります. 力をつけたい場合は, 解こうとしてから見ることを勧めます.

演習1-1: 回帰

□ 1変数の回帰問題

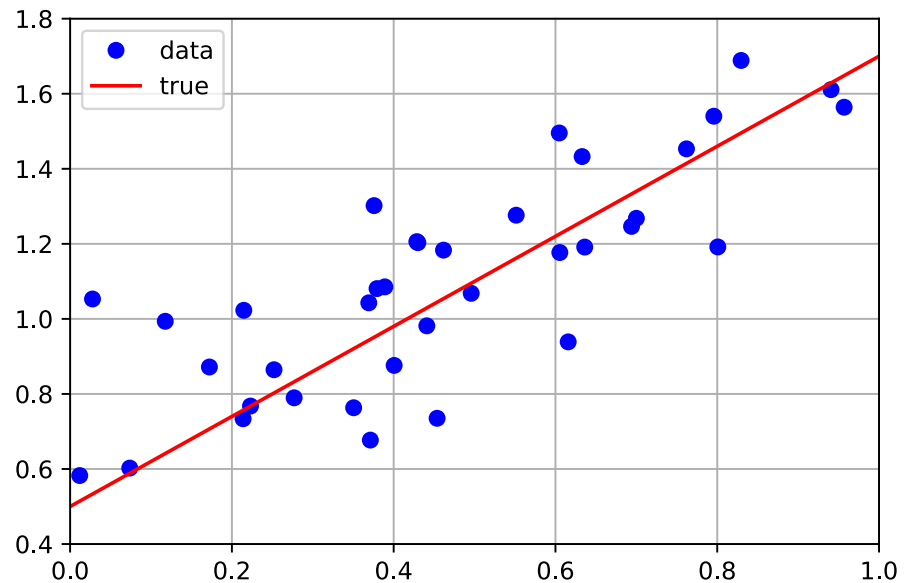
□ データ点群 $\{x_n, y_n\}$ から直線 $y = ax + b$ を推定したい

□ 修得してほしい概念:

□ 機械学習のやりたいこと

□ ロス関数

□ ニューラルネットの設計



演習1-1: ニューラルネットの雑な理解

□ ニューラルネットとは
単純な演算ユニットを沢山集めて作られた計算機

□ 構成ユニット

□ 積和計算と非線形活性による変換関数

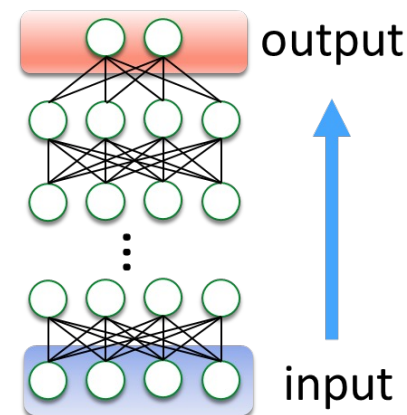
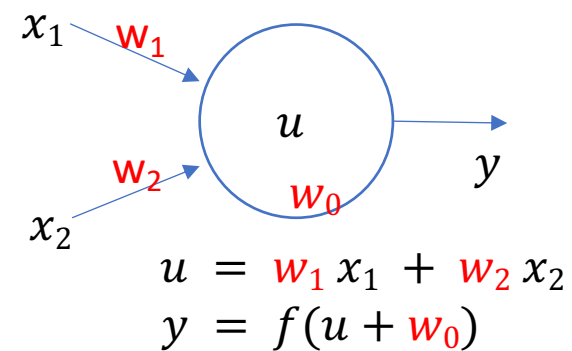
□ ノードとエッジによる計算

□ これが大量につながってできると

□ 多数の素子による並列, 協調計算

□ 任意精度の関数近似機械

□ パラメータ w によっていろいろ化けれる

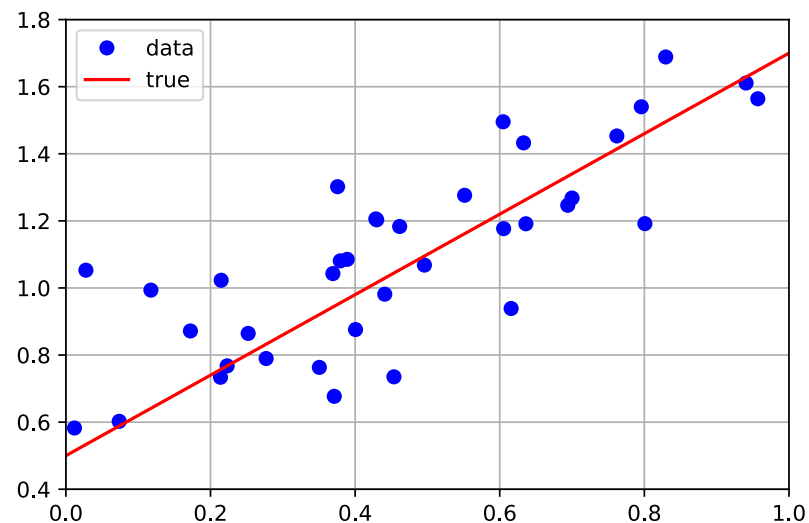
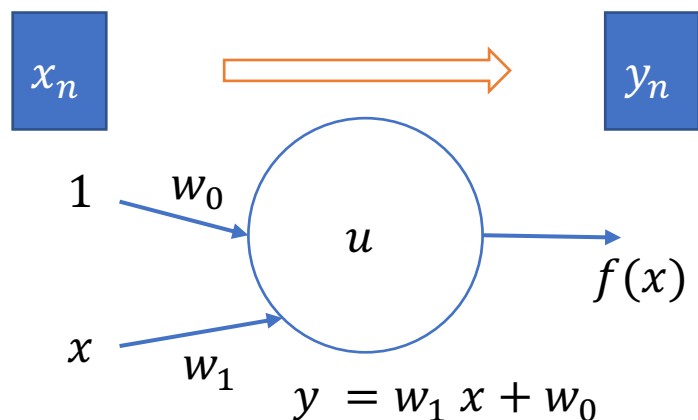


演習1-1: ロス関数

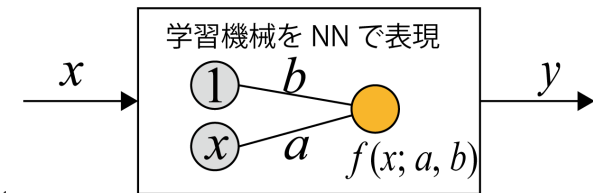
- 解きたい課題は, x_n を入力したときに y_n に近い出力を吐き出す関数 $f(x)$ を構成すること
- デザインとしては下のようなものを考えればよい

$$J(w) = \frac{1}{N} \sum_n (y_n - f(x_n))^2$$

\uparrow y 座標値 \uparrow NNの答え



演習1-1: PyTorch による実現(1)



- 普通はこんな簡単な問題に用いないが,
1 入力 1 出力, 非線形変換なしのニューラルネット

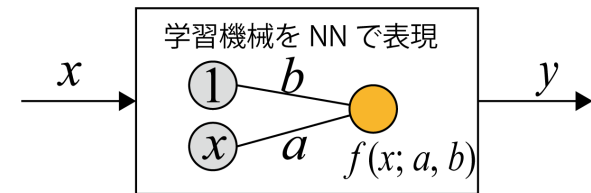
```
class LinearRegression(nn.Module):
    def __init__(self):
        super().__init__() # おまじない
        self.layer = nn.Linear(1, 1, bias=True) # 構造の記述

    def forward(self, x):
        '''順方向の計算, 入力xを出力y へ''' # 線形変換のみで記述
        y = self.layer(x)
        return y
```

- ロスは平均二乗誤差なので `MSELoss()` 関数を使う

```
loss = nn.MSELoss()
```

演習1-1: PyTorch による実現(2)



- 損失関数（ロス）は平均二乗誤差 `MSELoss()` 関数を使う

```
loss_func = nn.MSELoss()
```

- 最適化関数に, モデルを渡して

```
optimizer = torch.optim.SGD(model.parameters(), ...)
```

- 繰り返し計算で損失関数を下げるパラメータを探索

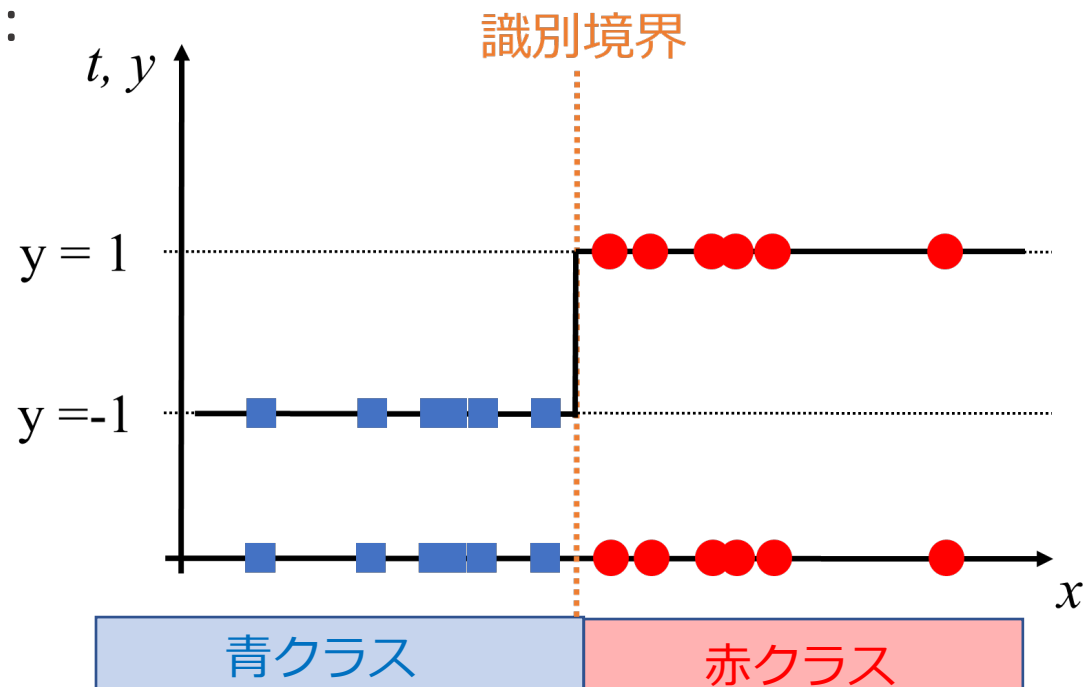
```
for epoch in range(10000):  
    optimizer.zero_grad()  
    outputs = model(inputs) # 現在のモデル出力  
    loss = loss_func(outputs, targets) # ズレを算出  
  
    loss.backward() # ズレを小さくする方向を算出  
    optimizer.step() # パラメータ更新
```

演習1-2: パターン認識課題 (1変数)

- 1変数のパターン認識
- ラベル付データ点群 $\{x_n, y_n\}$ から, **境界**を推定したい

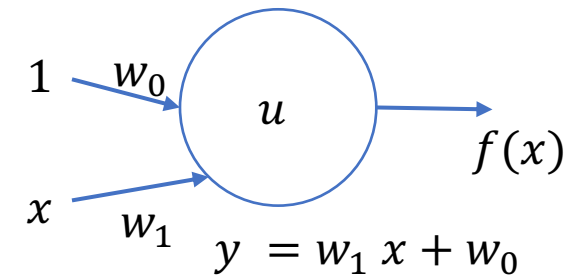
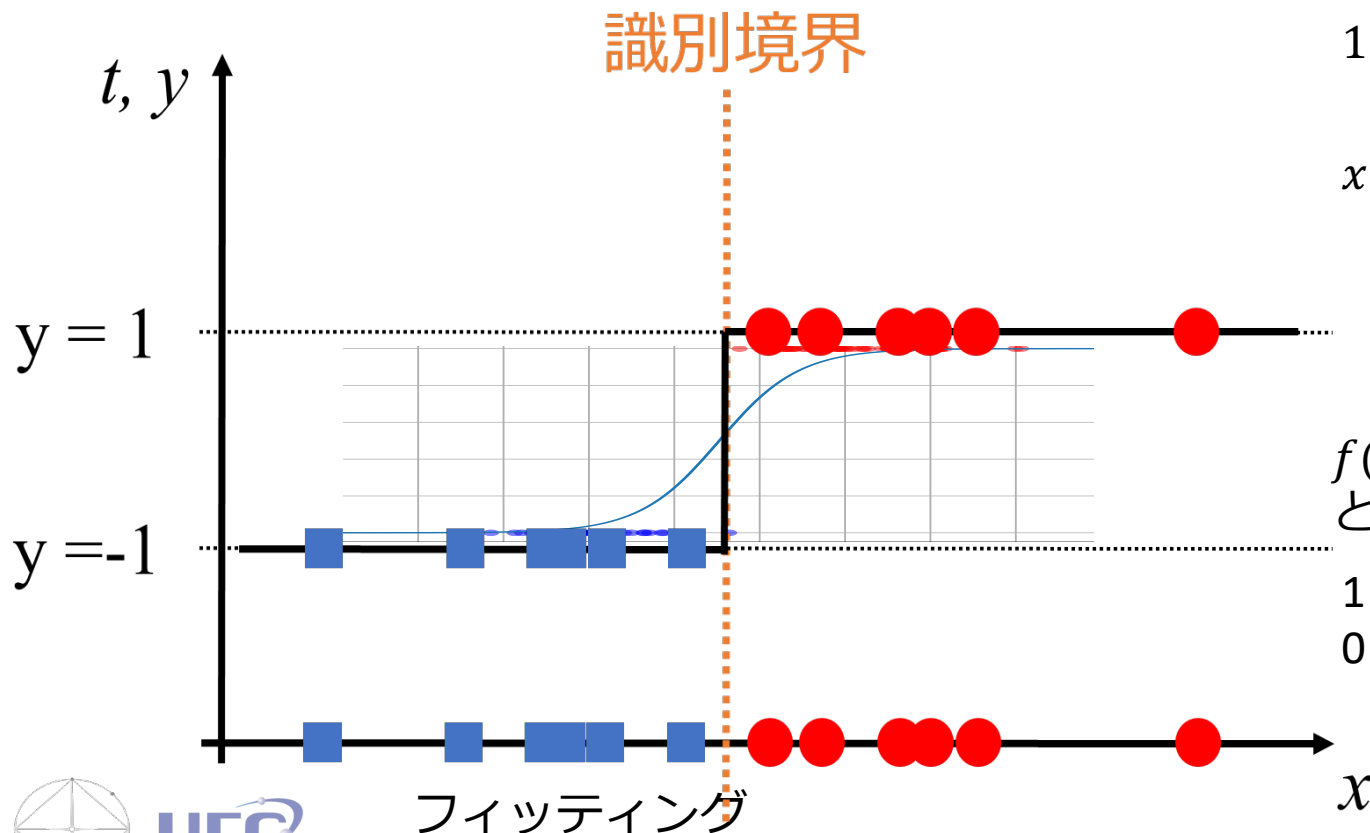
- 修得してほしい概念:

- 活性化関数
- ロジスティック回帰



演習1-2: 活性化関数

- 欲しいのは x_n を入力したときに
妥当なラベル y_n を吐き出す関数 $f(x)$



$f()$ をロジスティック関数としてフィットする

1 に近ければ赤,
0 に近ければ青と判断

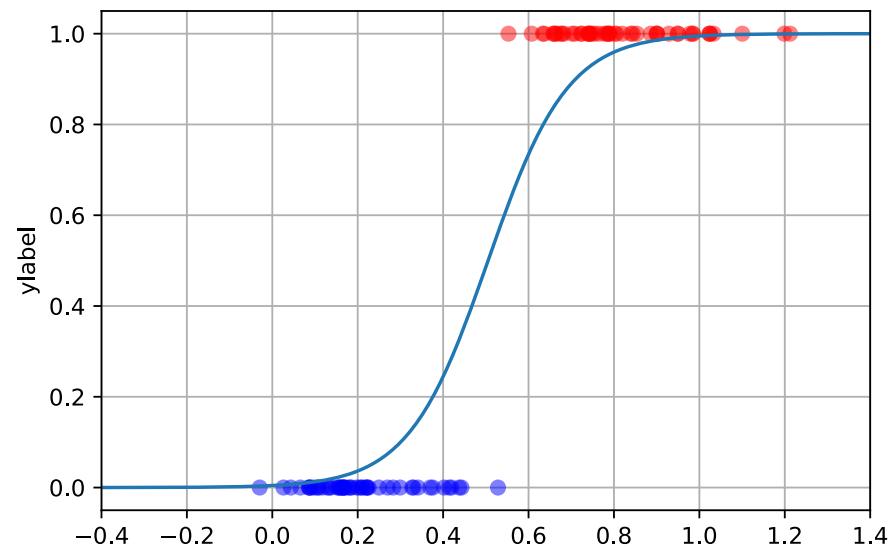
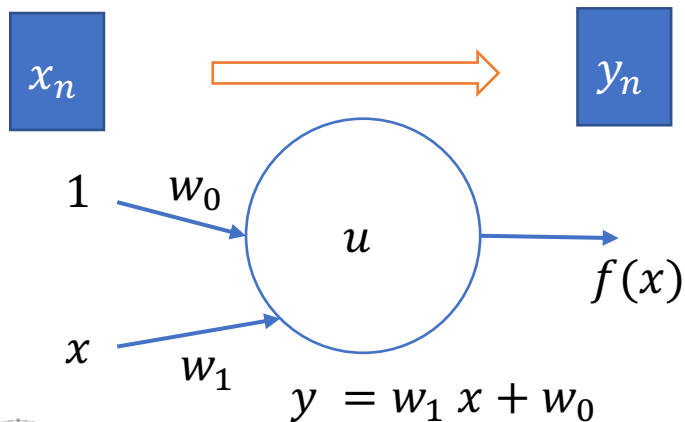
演習1-2: ロス関数

- クラス分類の場合は交差エントロピー関数をロス関数に使うのが一般的

$$J(\mathbf{w}) = -\frac{1}{N} \sum_n y_n \log f(x_n) + (1 - y_n) \log(1 - f(x_n))$$

y 座標値

NNの答え

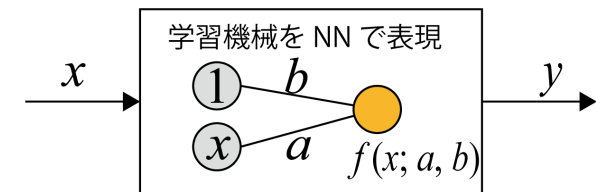


演習1-2: PyTorch による実現

- 普通はこんな簡単な問題に用いないが,
1 入力 1 出力のニューラルネット

```
class LogisticRegression(nn.Module):
    def __init__(self):
        super().__init__()
        self.layer = nn.Linear(1, 1, bias=True)
        self.f = nn.Sigmoid () # 活性化関数

    def forward(self, x):
        '''順方向の計算, 入力xを出力y へ'''
        y = self.layer(x)
        return self.f(y)
```

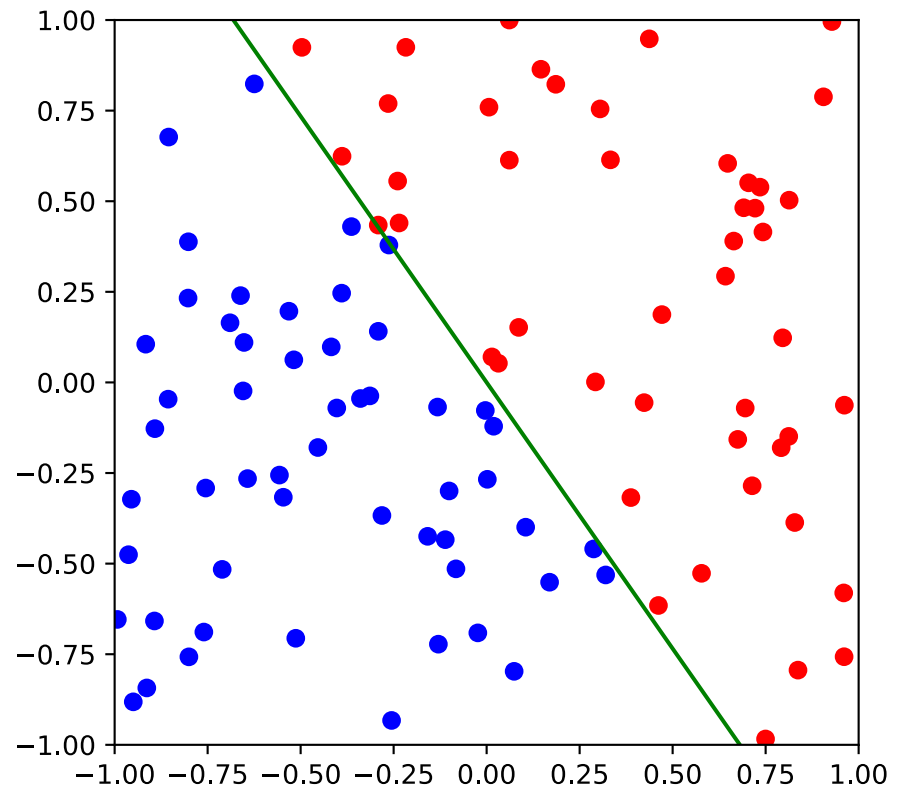


- ロスは交差エントロピー関数

```
loss_func = nn.BCELoss()
```

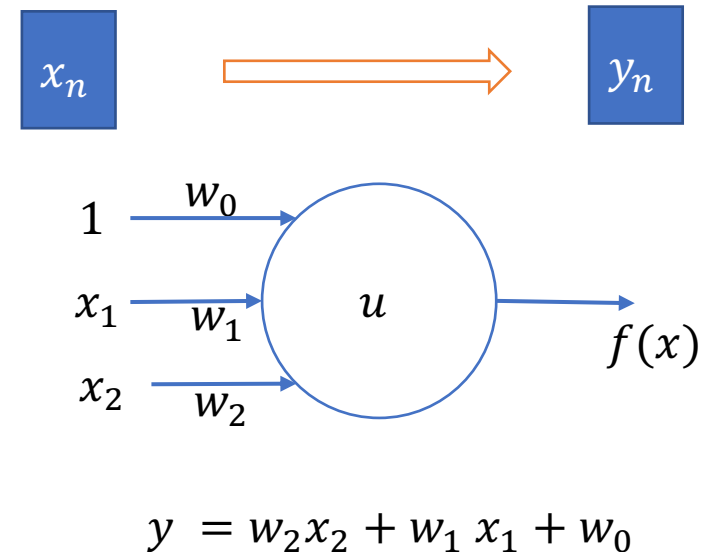
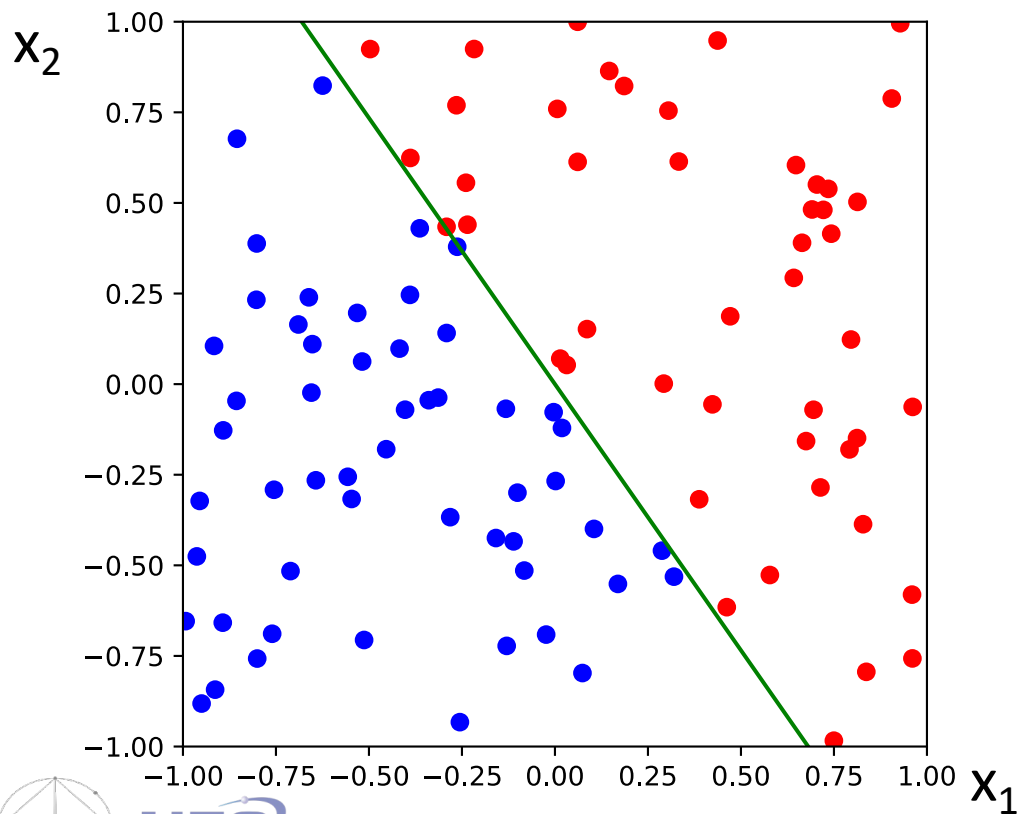
演習1-3: パターン認識課題 (多変数)

- 多変数のパターン認識
- ラベルつきデータ点群 $\{x_n, y_n\}$ から, 境界を推定
 - x_n がベクトルになる
- 修得してほしい概念:
 - 多変数の場合への拡張



演習1-3: ニューラルネットの形状

- 欲しいのは x_n を入力したときに
妥当なラベル y_n を吐き出す関数 $f(x)$



演習1-3: PyTorch による実現

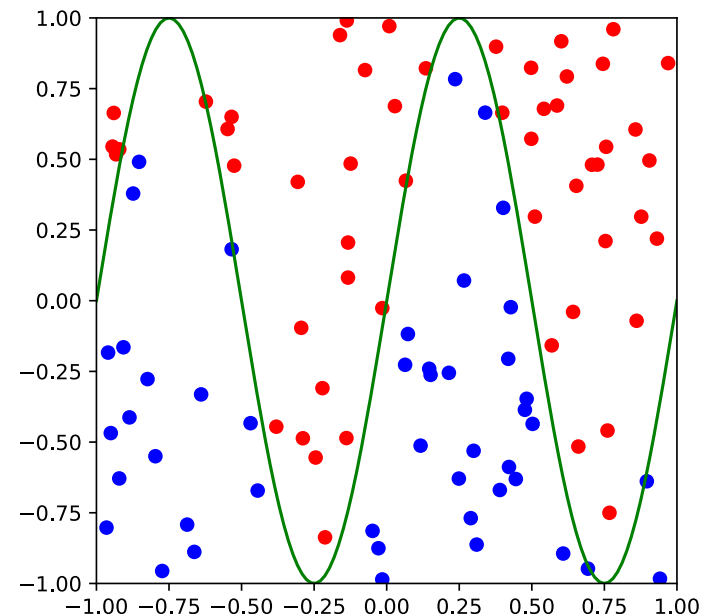
- 普通はこんな簡単な問題に用いないが,
2入力 1 出力のニューラルネット

```
class LogisticRegression(nn.Module):  
    def __init__(self):  
        super().__init__()  
        self.layer = nn.Linear(2, 1, bias=True)  
        self.f = nn.Sigmoid ()  
  
    def forward(self, x):  
        '''順方向の計算, 入力xを出力y へ''' # 線形変換のみで記述  
        y = self.layer(x)  
        return self.f(y)
```

- ロスは交差エントロピー

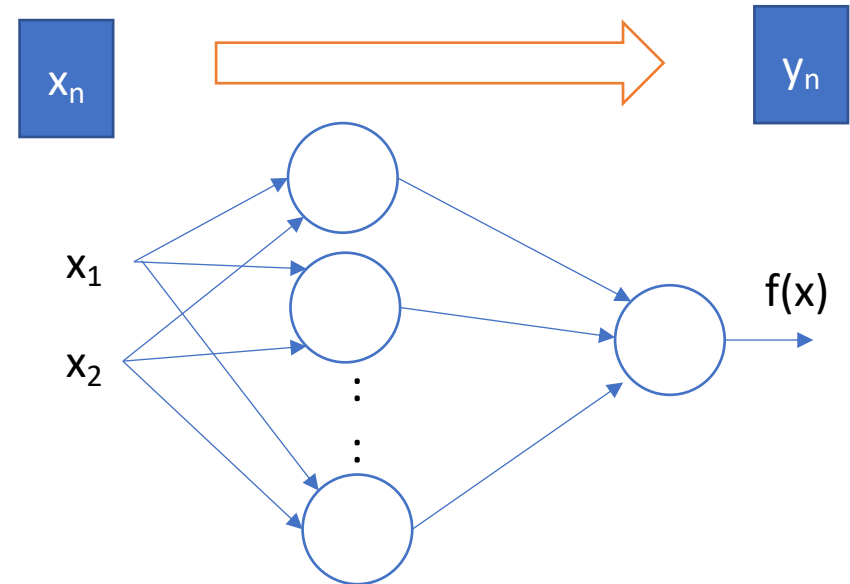
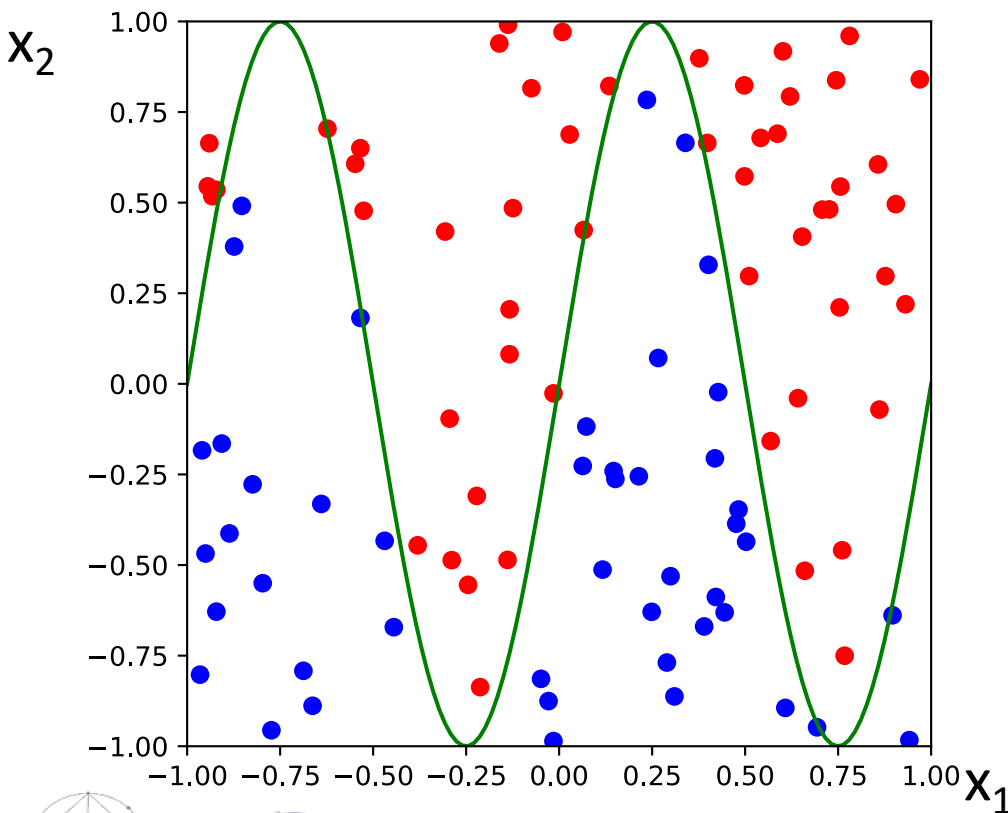
演習1-4: パターン認識課題: MLP

- 2変数のパターン認識
- ラベルつきデータ点群 $\{x_n, y_n\}$ から, 境界を推定
 - 境界が複雑で直線で分離出来ない
- 修得してほしい概念:
 - 多層化によってできることの確認



演習1-4: ニューラルネットの形状

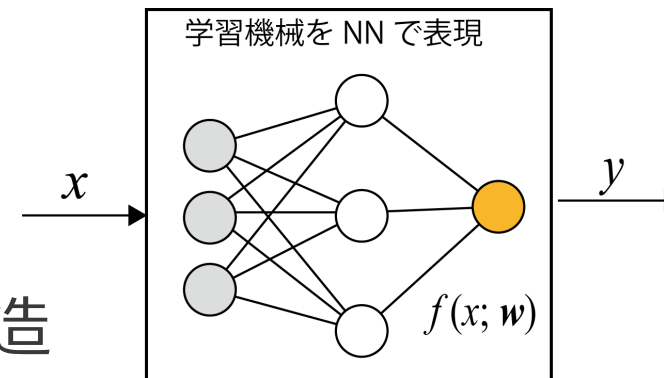
□ 欲しいのは x_n を入力したときに
妥当なラベル y_n を吐き出す関数 $f(x)$



直線では分離できなさそうなので
中間層を増やす

演習1-4: PyTorch による実現

- 中間層があるので MLP となる
下記は (2, 15, 1) 個の素子からなる構造

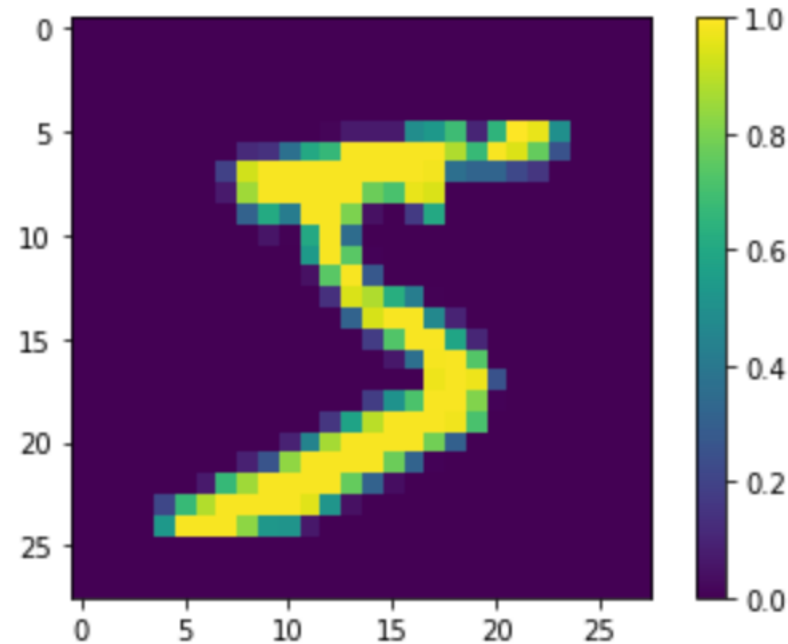


```
class LogisticRegression(nn.Module):
    def __init__(self):
        super().__init__()
        self.l1 = nn.Linear(2, 15, bias=True)
        self.l2 = nn.Linear(15, 1, bias=True)
        self.relu = nn.ReLU ()
        self.sigmoid = nn.Sigmoid()
    def forward(self, x):
        '''順方向の計算, 入力xを出力y へ''' # 線形変換のみで記述
        z = self.relu(self.l1(x))
        y = self.sigmoid(self.l2(z))
        return y
```

- ロスは交差エントロピー

演習1-5: MNISTをロジスティック回帰

- 2変数のパターン認識
- ラベルつきデータ点群 $\{x_n, y_n\}$ から, 境界を推定したい
- 修得すべき概念:
 - MNISTの取扱
 - 多クラス分類



28 x 28 = 784 次元の入力

課題 1

- 多層パーセプトロン(MLP)を PyTorch を用いて実現し, MNISTデータセットの識別性能を調査する.
- 与えられた Jupyter notebook を用いて答える.
- 深層学習を触ったことがあるなら一度は通る途です.
- いきなり答えがかけれるヒトはさっくり進めてもらって構いません.
 - ただし演習課題も解くこと.