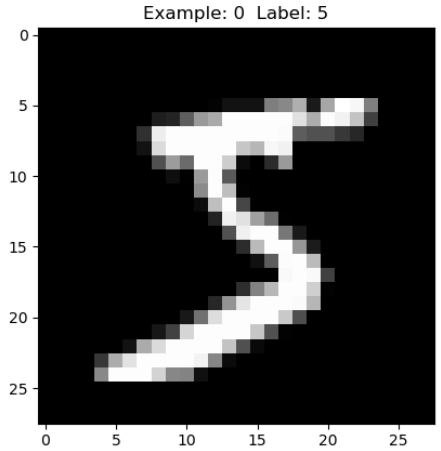
2020･07･15　ゼミ資料

ゼロから作るDeepLearning
第4章

まさぁぁ

データから学習する

何をもって “5” と認識するか?



- 非線形な分離問題は何千ものパーセプトロンを
  自分で設定しないといけない → ほぼ不可能

- 解決のアイデアの1つは特徴量を抽出して特徴量のパターンを機械学習する.

- 今回はニューラルネットワークによるアプローチをする.
  → 人が介在せず, 更にすべての問題を同じ流れで解ける.

データから学習する

訓練データとテストデータ

・汎化能力を正しく評価するために訓練データとテストデータに分ける

・　汎化能力を評価することは過学習をしているかのチェックにつながる.

損失関数

損失関数

- ニューラルネットの性能の悪さを示す指標.

- 損失関数の性能の悪さを最小化することは
  ニューラルネットワークの性能の良さを最大化することと同値.

- 2乗和誤差や交差エントロピー誤差などが用いられる.

- なぜ損失関数を使うのかは後述する.

損失関数

二乗和誤差 (mean squared error)

y_k をニューラルネットワークの出力, t_k を教師データ, k をデータの次元数とする.

$$E = \frac{1}{2}\sum_k (y_k - t_k)^2$$

で表されるもの.

損失関数

実験

```
import numpy as np
def mean_squared_error(y, t):
    return 0.5 * np.sum((y - t)** 2)

if __name__ == '__main__':
    t = np.array([0, 0, 1])
    y = np.array([0.1, 0.2, 0.7])
    print(mean_squared_error(y, t))
    # 0.07000000000000002

    y = np.array([0.7, 0.2, 0.1])
    print(mean_squared_error(y, t))
    # 0.6699999999999999
```

正解ラベルに特異的だと損失関数の値が小さいことがわかる.

損失関数

交差エントロピー誤差 (cross entropy error)

y_k をニューラルネットワークの出力, t_k を教師データ, k をデータの次元数とする.

$$E = -\sum_k t_k \log(y_k)$$

で表されるもの.

交差エントロピーを理解してみる.
http://yaju3d.hatenablog.jp/entry/2018/11/30/225841

損失関数

分類問題において $k = i$ に限り $t_k = 1$ とすると,

$$E = -\log(t_i)$$

が成立.

- 実装上の注意: $y_k = 0$ となるkが存在するとバグる.
  yに微小な値を足してあげる.

損失関数

実際問題では訓練データ全体の損失関数を最小化することが目的なので，
訓練データ数をNとして，交差エントロピー誤差を以下のように定める．

$$E = -\frac{1}{N}\sum_{n}\sum_{k} t_{nk}\log(y_{nk})$$

ここでt_{nk}はn個目のデータのk番目である．

損失関数

バッチ学習

- 前述のEを求めるための計算量はlogの計算量が$O(1)$と仮定すると,
  全体で$O(NK)$である. これを何度も計算するため, もう少し軽い処理をしたい.

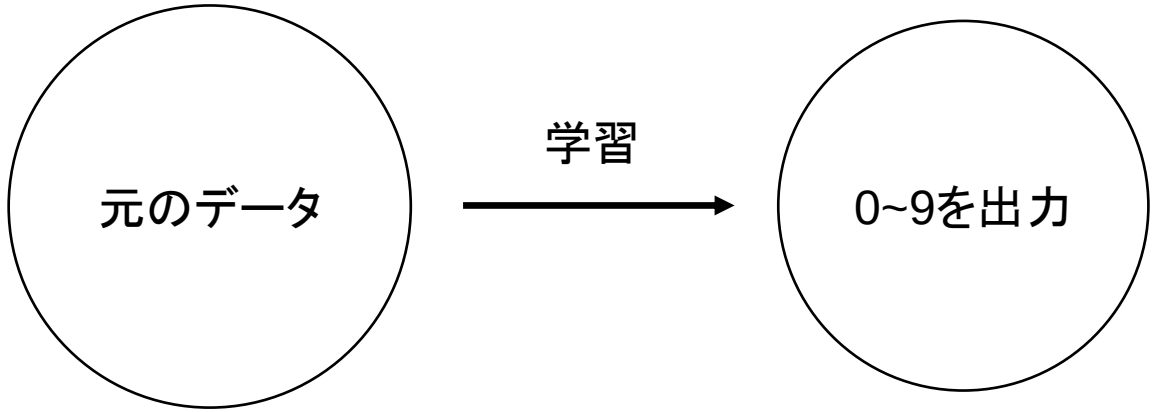
- バッチ学習…データの中から一部をランダムに取り出し
  全体の近似として利用する.
  → ミニバッチ学習 という.

$$E = -\frac{1}{|S|}\sum_{n\in S}\sum_{k} t_{nk}\log(y_{nk})$$

損失関数

損失関数を設定する理由

- 学習パラメータを少し動かしても
  出力結果は変わらない.
- 損失関数は少し動かすと値が変わるので
  微分ができそう.
- そういや微分で最小値が求まった気がする.



これ以上になく簡略化した図

数値微分

数値微分

- 微分

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

- 数値微分

hを十分小さい数として（あまりに小さいと丸め誤差で死ぬ）

$$f'(x) = \frac{f(x+h) - f(x)}{h}$$ （前方差分）

数値微分

うまい実装

$f'(x)$ が存在する時,

$$\lim_{h\to 0}\frac{f(x+h)-f(x-h)}{h}=\lim_{h\to 0}\left(\frac{f(x+h)-f(x)}{h}+\frac{f(x-h)-f(x)}{-h}\right)=2f'(x)$$

を利用して

$$f'(x)=\frac{f(x+h)-f(x-h)}{2h}\qquad\text{（中心差分）}$$

とすると, 右からの差分と左からの差分の平均を取るので収束が速い. (多分)

数値微分

数値微分の例

省略　(言われたことをちゃんと実装する.)

偏微分

偏微分

簡単のため2変数関数で記す. (3変数以上も同様)

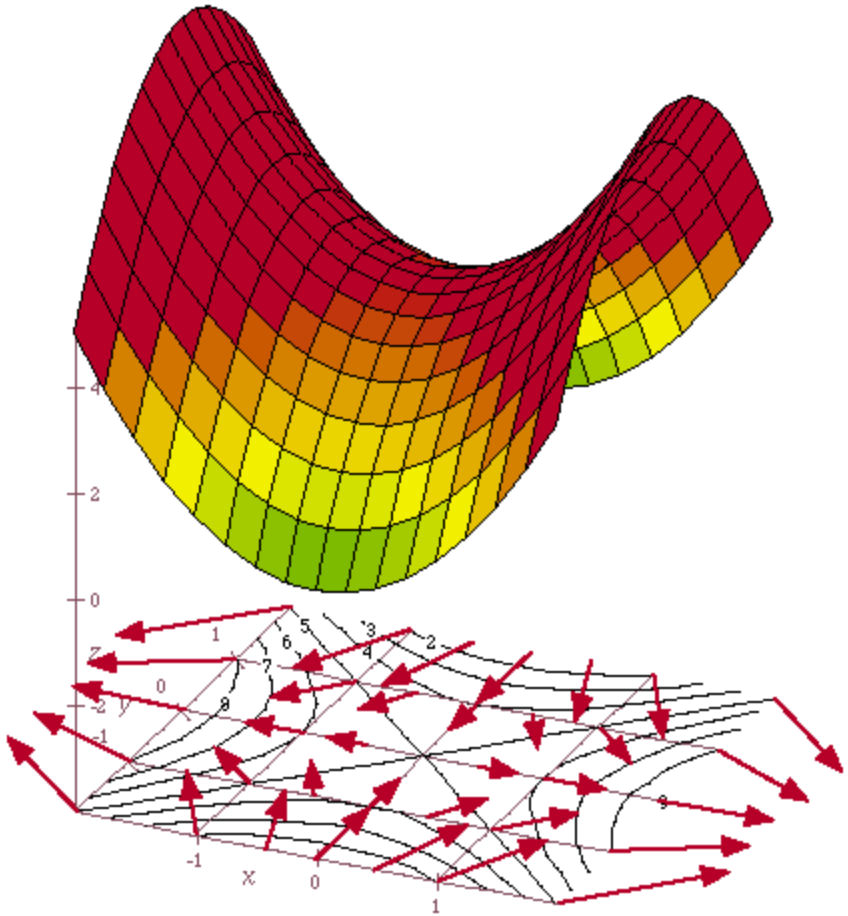
$$\frac{\partial f}{\partial x_0} = \lim_{h \to 0} \frac{f(x_0 + h, x_1) - f(x_0, x_1)}{h} \qquad \frac{\partial f}{\partial x_1} = \lim_{h \to 0} \frac{f(x_0, x_1 + h) - f(x_0, x_1)}{h}$$

勾配

勾配

$$grad(f) = \left(\frac{\partial f}{\partial x_0}, \frac{\partial f}{\partial x_1} \right)$$

- $grad(f)$は最大傾斜方向を指す. (教科書は誤り)
- これを使って局所解を予測する.



http://www3.u-toyama.ac.jp/math/kyouzai/gradient/index.html

勾配法

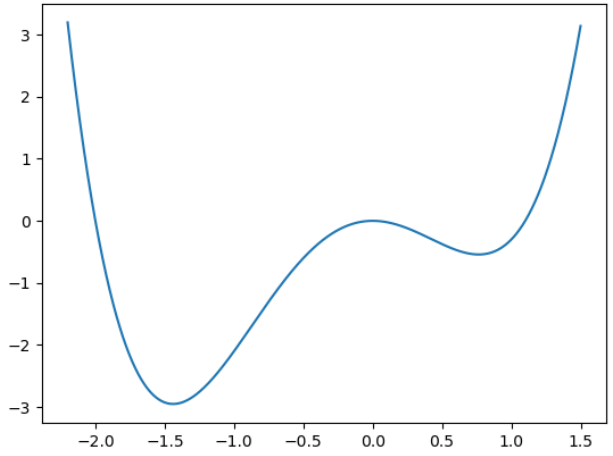
勾配法

- 局所最小解を求めるアルゴリズム.

アルゴリズム

$$(x_0, x_1) = (x_0, x_1) - \eta * grad(f)$$

- 大域的な最小解を求めれる保証はない.



勾配法

学習率

勾配法

$$(x_0, x_1) = (x_0, x_1) - \eta * grad(f)$$

- ηのことを学習率という. (パラメータ更新の速さ)
- 学習率は大きすぎると収束しないし,
  小さすぎると収束までのstep数が大きくなるため注意が必要.
- 学習率を変更しながらうまく学習できているか確認するのが一般的
- 重みやバイアスのように自動で取得されるパラメータと異なり,
  手動で設定するパラメータのことをハイパーパラメータという.

My Question : 学習率は前もって何らかの値に決める必要がある

勾配法

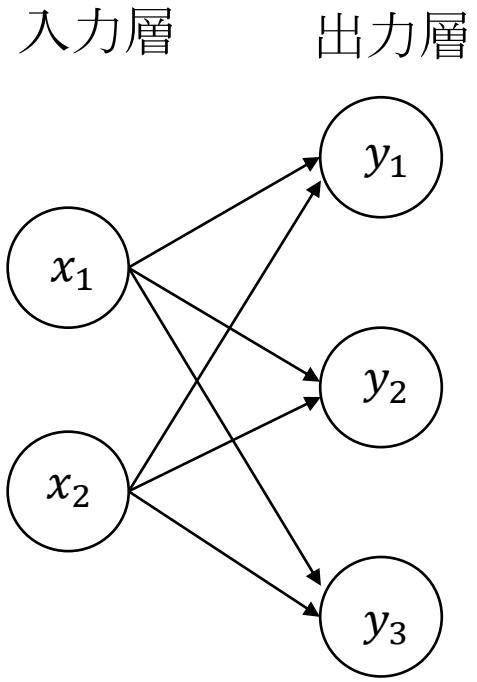
ニューラルネットワークに対する勾配

・Wの各成分に対して勾配を求める (今回だと6次元ベクトルと思う)

$$W = \begin{pmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{pmatrix}$$

$$\frac{\partial L}{\partial W} = \begin{pmatrix} \frac{\partial L}{w_{11}} & \frac{\partial L}{w_{12}} & \frac{\partial L}{w_{13}} \\ \frac{\partial L}{w_{21}} & \frac{\partial L}{w_{22}} & \frac{\partial L}{w_{23}} \end{pmatrix}$$

L は損失関数



学習アルゴリズムの実装

実装の方針

- ニューラルネットワークでパラメータ（重みとバイアス）を学習する.

for _ in range(繰り返す回数):

1. ミニバッチを取り出す.
2. 勾配を算出.
3. （確率的)勾配降下法によりパラメータを更新.

学習アルゴリズムの実装

mnistを使った2層ニューラルネットワーク

- 省略　(コードはやや難解なためわからなかったら質問して)

  過学習もなく, 損失関数も小さくなっていてうまくいっていることがわかる.

まとめ

- 損失関数を使った勾配降下法による学習を学んだ.

キーワード：訓練データ, テストデータ, 過学習, 損失関数, 二乗和誤差, 交差エントロピー誤差, ミニバッチ, 数値微分, 勾配法, ハイパーパラメータ, エポック