ニューラルネットワークにおける過学習に関する分析
- On the Over-fitting in the Neural Network -

鷲野朋広
Tomohiro Washino
甲南大学大学院 自然科学研究科 知能情報学専攻
Graduate School of Natural Science, Konan University *
高橋正
Tadashi Takahashi
甲南大学 知能情報学部 知能情報学科
Department of Intelligence and Informatics, Konan University †

Abstract

第 1 に, フィッシャーのアイリスデータに対してニューラルネットワークを作成し, 2 種類 (setosa, versicolor) を学習させる. 次に, 学習済みのニューラルネットワークにおいて 3 種類 (setosa, versicolor, virginica) を訓練データ, 1 種類 (virginica) をテストデータとして学習させる. このときテストデータの影響を過剰に受けてしまう状態について分類表を作成する. 第 2 に, 学習モデル (中間ユニット数が 1000 である 3 層パーセプトロン) を作成し, 真の分布に従う訓練データを学習させる. このときニューラルネットワークの出力が訓練データを過剰に近似して過学習が起きる. この状態について訓練データとテストデータに対する損失を比較して調べる. また, 訓練データとテストデータに対して, 学習後のニューラルネットの出力をグラフで表す. 次に, 学習モデル (中間ユニット数が 2 である 3 層パーセプトロン) を作成し, 訓練データを学習させる. このときニューラルネットワークの出力が訓練データを過剰に近似せず過学習が起こらない. この状態について学習の過程で変化する RMS 重みをグラフで表す. 最後に学習後の重みの収束値が真の分布を実現する学習モデルのパラメータ集合を満たすことを確かめる.

Abstract

First, let us construct neural network for the Fisher's iris data, we submit two species of iris flowers (setosa, versicolor). Next, let us extract trained network, we submit training data about two species of iris flowers (setosa, versicolor) and test data about one species of iris flowers (virginica). Then, we consider a state excessively affected by test data and create a classification table. Second, let a statistical model be a three-layer neural network with 1000 hidden units, we submit training data about true density function. Then, an output of the neural network is excessively approximated by training data. We show that over-fitting has occurred by comparing training and generalization error. We plot trained neural network on training data and test data. Next, let the statistical model be a three-layer neural network with 2 hidden units, we submit training data . Then, an output of the neural network is not excessively approximated by training data. We show that over-fitting has not occurred and plot RMS weights that evolve during submitting. Finally, we show that weights at the end of submitting satisfy a set of parameters which true density function is realizable by the statistical model.

*d1523001@s.konan-u.ac.jp
†takahasi@konan-u.ac.jp

1　はじめに

ニューラルネットワークで「教師あり学習」をさせると, 度々発生するものに「過学習 (over-fitting)」がある.

定義 1 (真の分布, 学習モデル 事前分布)
$\mathbb{R}^N$ 上の確率密度関数 $q(x)$ に対し, $q(x)$ に従う n 個の独立な確率変数 $X^n = (X_1, X_2, ..., X_n)$ を例という. 例を発生している確率分布 $q(x)$ を真の分布という. $\mathbb{R}^N$ 上の x の確率密度関数 $p(x|w)$ を学習モデルという. また, $\mathbb{R}^d$ 上の w の確率密度関数 $\varphi(w)$ を事前分布という.

定義 2 (損失関数)
真の分布を推測するため, 非負の実数列 $\{a_n\}$ に対して損失関数を次のように定める.

$$R_n(w) := -\sum_{i=1}^{n} \log p(X_i|w) - a_n \log \varphi(w).$$

定義 3 (正規化された損失関数)
このとき経験エントロピー $S_n := -\frac{1}{n}\sum_{i=1}^{n} \log q(X_i)$ を用いて $R_n(w) = R_n^0(w) + nS_n$ と表されるため, 対数尤度比関数 $f(x, w) := \log \frac{q(x)}{p(x|w)}$ に対して正規化された損失関数を次のように定める.

$$R_n^0(w) := \sum_{i=1}^{n} f(X_i, w) - a_n \log \varphi(w).$$

ここで $R_n^0(w)$ を最小にするパラメータを $\hat{w}$ を求め, $p(x|\hat{w})$ を推測結果の確率密度関数とする. 推測法として $a_n = 0$ のとき $\hat{w}$ を最尤推定法, $a_n = 1$ のとき事後確率最大化推定法という [1, 2].

定義 4 (経験損失, 汎化損失)
このとき汎化損失 $K(\hat{w})$ と経験損失 $K_n(\hat{w})$ を次で定める.

$$R_g = K(\hat{w}) := \int q(x) \log \frac{q(x)}{p(x|\hat{w})} dx,\ R_t = K_n(\hat{w}) := \frac{1}{n}\sum_{i=1}^{n} \log \frac{q(X_i)}{p(X_i|\hat{w})}.$$

事前分布のサポート $\mathrm{supp}(\varphi)$ はコンパクト, 対数尤度比関数 $f(x, w)$ は相対的に有限な分散をもつとする. 真の分布を実現するパラメータの集合 $W_0 := \{w \in \mathbb{R}^d | K(w) = 0\}$ が特異点を持つとき, $w_0 \in W_0$ として w_0 の近傍で特異点解消定理を適用させると, 解析多様体 M と解析写像 $g : M \to W$ が存在して $w = g(u)$ となる. このとき自然数 $k_1, k_2, \cdots, k_r$ に対して $K(g(u)) = u_1^{2k_1} u_2^{2k_2} \dots u_r^{2k_r}$, $(1 \le r \le d)$ と表される. また $\int a(x, u) q(x) dx = u_1^{k_1} u_2^{k_2} \dots u_r^{k_r}$ を満たす解析関数 $a(x, u)$ が存在して $f(x, g(u)) = a(x, u) u_1^{k_1} u_2^{k_2} \dots u_i^{k_i}$ と表される. このとき経験誤差関数 $K_n(g(u))$ は確率過程 $\xi_n(u) := \frac{1}{\sqrt{n}}\sum_{i=1}^{n}\{u^k - a(X_i, u)\}$ を用いて $nK_n(g(u)) = nu^{2k} - \sqrt{n}u^k\xi_n(u)$ と表される. ここで $u^{2k} := u_1^{2k_1} u_2^{2k_2} \dots u_r^{2k_r}$, $u^k := u_1^{k_1} u_2^{k_2} \dots u_r^{k_r}$ であり, $\xi_n(w)$ は正規確率過程 $\xi(w)$ に法則収束する. このとき正規化された損失関数は次のように表される.

$$\frac{1}{n} R_n^0(g(u)) = u^{2k} - \frac{1}{\sqrt{n}} u^k \xi_n(u) - \frac{a_n}{n} \log \varphi(g(u)).$$

次に $t \in \mathbb{R}^1$, $v = (v_1, v_2, \cdots, v_d) \in \mathbb{R}^d$ に対して $t = u^{2k}$, $v_i = \begin{cases} \sqrt{u_i^2 - \frac{k_i}{k_a} u_a^2} & (1 \le i \le r) \\ u_i & (r \le i \le d) \end{cases}$ とすると $v \in V := \{v = (v_1, v_2, \cdots, v_d) \in [0, 1]^d | v_1, v_2, \dots, v_d = 0\}$ が成り立つ.
よって写像 $u \in [0, 1]^d \to (t, v) \in T \times V$ を定める. このとき正規化された損失関数は次のように表される.

$$\frac{1}{n} R_n^0(g(t, v)) = t^2 - \frac{1}{\sqrt{n}} t \xi_n(t, v) - \frac{a_n}{n} \log \varphi(g(t, v)).$$

定理 5
対数損失関数が相対的に有限な分散をもつとする. $U_0 := \{u \in [0,1]^d | K(g(u)) = 0\}$ を満たす集合を最大にするパラメータ $\hat{u}$ を次で定める.

$$\hat{u} := \arg \max_{K(g(u))=0} \left\{ \frac{1}{4} \max\{0, \xi_n(u)\}^2 + \log \varphi(g(u)) \right\}.$$

このとき平均対数損失関数 $L(w) := -\int q(x) \log p(x|w) dx$ と経験対数損失関数 $L_n(w) := -\frac{1}{n} \sum_{i=1}^n \log p(X_i|w)$ に対して汎化損失と経験損失は次の挙動を持つ.

$$K(\hat{w}) = L(g(\hat{u})) - L(w_0) = \frac{1}{4n} \max\{0, \xi_n(\hat{u})\}^2 + o_p\left(\frac{1}{n}\right),$$

$$K_n(\hat{w}) = L_n(g(\hat{u})) - L_n(w_0) = -\frac{1}{4n} \max\{0, \xi_n(\hat{u})\}^2 + o_p\left(\frac{1}{n}\right).$$

最急降下法では t の部分は急速に最適値に近づくが v の部分はなかなか最適値に近づかない. 一般には t についての最適化は汎化損失を小さくするが v についての最適化は汎化損失を大きくする [3, 4].

定義 6 (過学習)
経験損失は単調に小さくなるが汎化損失は途中から大きくなる現象を過学習という.

2 定義

第 k 層に n 個のユニットがあるニューラルネットワークを考え, 右の図 1 に表す.

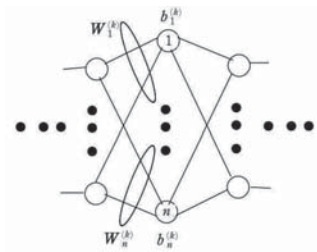

図 1: 線形層

定義 7 (線形層)
第 $k-1$ 層の出力ユニット $(y_1^{(k-1)}, y_2^{(k-1)}, \cdots, y_m^{(k-1)})$ 対して, 第 k 層の j 番目 $(1 \leq j \leq n)$ の入力ユニット $x_j^{(k)}$ を次で定める.

$$x_j^{(k)} := \sum_{i=1}^m w_{ji}^{(k)} y_i^{(k-1)} + b_j^{(k)} = \mathbf{w}_j^{(k)^T} \mathbf{y}^{(k-1)} + b_j^{(k)} = \tilde{\mathbf{w}}_j^{(k)^T} \tilde{\mathbf{y}}^{(k-1)}$$

ここで, $w_{ji}^{(k)}$ は第 $k-1$ 層の j 番目ユニットから第 k 層の i 番目ユニットへの重み, $b_j^{(k)}$ は第 k 層の j 番目ユニットのバイアスと定め,

ベクトル $\mathbf{w}_j^{(k)} = (w_{j1}^{(k)}, \cdots, w_{jm}^{(k)})$, $\mathbf{y}^{(k-1)} = (y_1^{(k-1)}, \cdots, y_m^{(k-1)})$, $\tilde{\mathbf{w}}_j^{(k)} = (\mathbf{w}_j^{(k)}, b_j^{(k)})$, $\tilde{\mathbf{y}}^{(k-1)} = (\mathbf{y}^{(k-1)}, 1)$ とする.

定義 8 (活性化関数)
第 k 層の j 番目の出力ユニットを $y_j^{(k)} := \varphi\left(x_j^{(k)}\right) = \tanh\left(x_j^{(k)}\right)$ で定める. ここで $\varphi(x) = \tanh(x)$ とし, 活性化関数という.

Mathematica を用いて次のように入力する [5].
学習後のネットグラフからニューラルネットワークの部分を取り出すために
$trainednet = NetReplacePart[NetExtract[results["TrainedNet"], "net"]]$ とする.
訓練データの損失の変化をグラフに書くために $results["LossEvolutionPlot"]$ とする.
テストデータについて最低平均損失を求めるために $results["LowestValidationRound"]$ とする.
テストデータについて最終ラウンドでの損失を求めるために $results["FinalValidationLoss"]$ とする.

3　アイリスデータによる過剰般化

「過剰般化 (over-generalization)」とは，新しく学習した内容に過剰に適合してしまい，既に学習した内容の正解率が低くなってしまう状態とする．

あやめのがく片長 (Sepal Length), がく片幅 (Sepal Width), 花びら長 (Petal Length) 花びら幅 (Petal Width) の 4 個の計測値から, setosa, versicolor, virginica という 3 種類の種 (Species) に分類するデータをアイリスデータといい,$\{4.7, 3.2, 1.3, 0.2\} \to \{"setosa"\}$ のように表す．

Mathematica に記録されている訓練データ 105 個, テストデータ 45 個を次のように入力して呼び出す．

$trainingData = ExampleData["MachineLearning","FisherIris","TrainingData"];$
$testData = ExampleData["MachineLearning","FisherIris","TestData"];$

Mathematica を用いて,$labels = Union[Values[trainingData]]$; と入力し，3 種類を 3 組に組分けする．

3 種類から 3 組を出力するネットエンコーダーを $enc = NetEncoder["Class", labels]$ と入力する．
確率ベクトルから 3 種類を出力するネットデコーダーを $dec = NetDecoder["Class", labels]$; と入力する．
確率ベクトルから 3 組を出力するネットデコーダーを $dec2 = NetDecoder["Class", 1, 2, 3]$; と入力する．

訓練データとテストデータ (出力を組とする) を次のように入力して定める．

$TrainingData := Normal[AssociationThread[Keys[trainingData] -> enc[Values[trainingData]]]]$
$TestData := Normal[AssociationThread[Keys[testData] -> enc[Values[testData]]]]$

部分データ (i, j 組), (i 組) を次のように入力して定める．

$trainset[i\_, j\_] := Select[TrainingData, Last[\#] == i || Last[\#] == j\&]$
$testset[i\_, j\_] := Select[TestData, Last[\#] == i || Last[\#] == j\&]$
$trainset[i\_] := Select[TrainingData, Last[\#] == i\&]$ $testset[i\_] := Select[TestData, Last[\#] == i\&]$

損失関数への入力 (長さデータ) と対象 (組) を，全体データ (1, 2, 3 組), 部分データ (i, j 組), (i 組) に対して, 次のように入力する．

$train[1, 2, 3] :=< "Input" -> Keys[TrainingData], "Target" -> Values[TrainingData] | >$
$train[i\_, j\_] :=< "Input" -> Keys[trainset[i, j]], "Target" -> Values[trainset[i, j]] | >$
$test[i\_, j\_] :=< "Input" -> Keys[testset[i, j]], "Target" -> Values[testset[i, j]] | >$
$test[i\_] :=< "Input" -> Keys[testset[i]], "Target" -> Values[testset[i]] | >$

3.1　定義

定義 9 (ソフトマックス関数)
第 k 層の j 番目の出力ユニットを $y_j^{(k)} := \varphi\left(x_1^{(k)}, x_2^{(k)}, \cdots, x_n^{(k)}\right) = \dfrac{\exp\left(x_j^{(k)}\right)}{\sum_{i=1}^{n} \exp\left(x_i^{(k)}\right)}$ で定める．

ここで $\varphi(x_1, x_2, \cdots, x_n)$ とし，ソフトマックス関数という．

定義 10 (クロスエントロピー損失)
$t_i = 0 (i \neq k)$, $t_k = 1$ を満たす $t = (t_1, t_2, \cdots, t_n), y = (y_1, y_2, \cdots, y_n)$ に対して，クロスエントロピー損失を $E := -\sum_{i=1}^{n} t_i \log y_i$ で定める．

Mathematica を用いて次のように入力する [5].
正しく分類された割合を求めるために $measurements["Accuracy"]$ とする．
分類表を表示させるために $measurements["ConfusionMatrixPlot"]$ とする．

3.2　ニューラルネットワーク, ネットグラフの作成

入力が 4 次元ベクトル, 出力が組である 4 層ニューラルネットワーク（線形層, tanh, 線形層, ソフトマックス層）を作成するため, Mathematica を用いて次のように入力すると, 以下の図 2 の左側に出力される.

$$\begin{aligned} parameterNet = \quad & NetChain[\{LinearLayer[5], ElementwiseLayer[Tanh], \\ & LinearLayer[3], SoftmaxLayer[]\}, "Input" -> 4, "Output" -> dec] \end{aligned}$$

入力 (4 次元ベクトルをニューラルネットワークに通した出力値) と, 対象 (組) のクロスエントロピー損失を求める. Mathematica を用いて次のように入力してネットグラフを作成すると, 以下の図 2 の右側に出力される.

$$\begin{aligned} trainingNet = \quad & NetGraph[<|"net" -> net, "loss" -> CrossEntropyLossLayer["Index"]|>, \\ & \{NetPort["Input"] -> "net" -> NetPort["loss", "Input"], \\ & NetPort["Target"] -> NetPort["loss", "Target"]\}] \end{aligned}$$

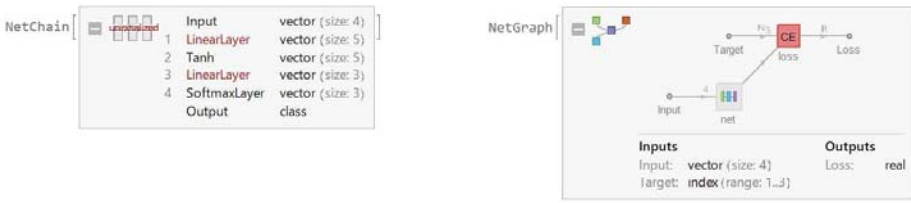

図 2: ニューラルネットワーク, ネットグラフ

3.3　アイリスデータ (1,2 組) の学習

バッチサイズを 35 として, 最大 500 ラウンド学習する中でテストデータ (1,2 組) の損失が最小となるラウンド数で学習を終えるように, 訓練データ (1,2 組) を学習させるために次のように入力する.

$$\begin{aligned} results = \quad & NetTrain[trainingNet, train[1,2], All, ValidationSet -> test[1,2], \\ & BatchSize -> 35, MaxTrainingRounds -> 500] \end{aligned}$$

学習結果の要約と, 損失の変化が図 3 の左側に出力される.

学習後のネットを trainednet と定めて, 次のように入力しテストデータ (1,2 組) の分類測度を求める.

$$measurements = ClassifierMeasurements[trainednet, testset[1,2]]$$

正確さは 1 であり分類測度と分類表が図 3 の右側に出力される.

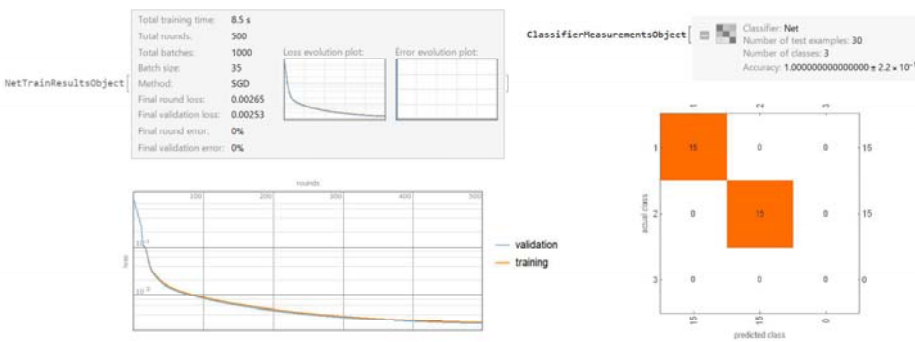

図 3: アイリスデータ (1,2 組) の学習, 分類

3.4　アイリスデータ (1,2,3 組) の学習

学習後のニューラルネットワークにおいて, バッチサイズを 35 として訓練データ (1,2,3 組) の学習をさせる. テストデータ (3 組) の損失が最小となるラウンド数で学習を終える.

ラウンド数が 8 のとき次のように入力する.

$$results1 = NetTrain[results["TrainedNet"], train[1,2,3], All,$$
$$ValidationSet- > test[3], BatchSize- > 35, MaxTrainingRounds- > 8]$$

損失が最小であるラウンド数は 8 であり, 学習結果の要約と, 損失の変化が図 4 の左側に出力される.

学習後のネットを trainednet1 と定めて, 次のように入力しテストデータ (1,2,3 組) の分類測度を求める.

$$measurements = ClassifierMeasurements[trainednet1, testset[1,2,3]]$$

正確さは 0.666667 であり分類測度と分類表が図 4 の右側に出力される.

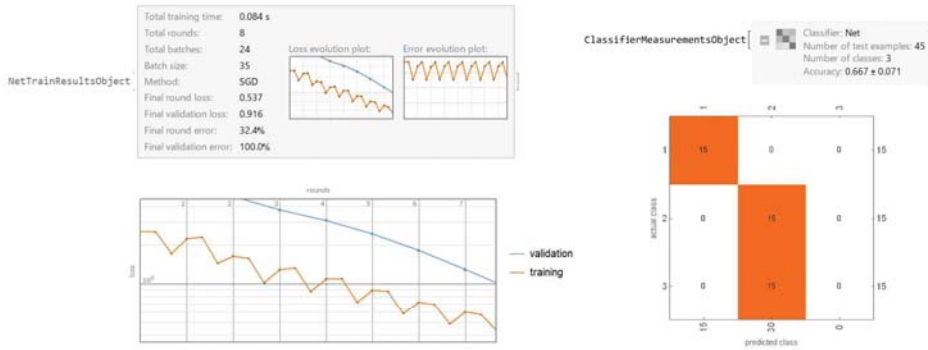

図 4: アイリスデータ (1,2,3 組) の学習, 分類

ラウンド数が 30 のとき次のように入力する.

$$results2 = NetTrain[results["TrainedNet"], train[1,2,3], All,$$
$$ValidationSet- > test[3], BatchSize- > 35, MaxTrainingRounds- > 30]$$

損失が最小であるラウンド数は 13 であり, 学習結果の要約と, 損失の変化が図 5 の左側に出力される.

学習後のネットを trainednet2 と定めて, 次のように入力しテストデータ (1,2,3 組) の分類測度を求める.

$$measurements = ClassifierMeasurements[trainednet2, testset[1,2,3]]$$

正確さは 0.666667 であり分類測度と分類表が図 5 の右側に出力される.

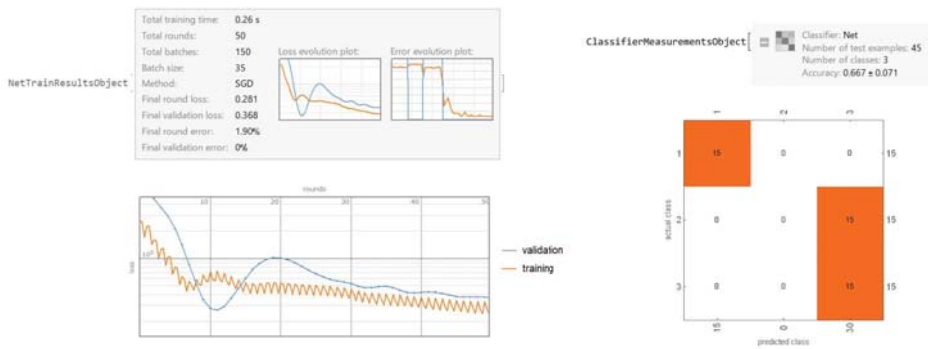

図 5: アイリスデータ (1,2,3 組) の学習, 分類 (過剰般化)

学習後のニューラルネットワークにおいて, バッチサイズを 35 として訓練データ (1,2,3 組) の学習をさせる. テストデータ (2,3 組) の損失が最小となるラウンド数で学習を終える.

ラウンド数が 100 のとき次のように入力する.

$$results3 = NetTrain[results2["TrainedNet"], train[1,2,3], All,$$
$$ValidationSet- > test[2,3], BatchSize- > 35, MaxTrainingRounds- > 100]$$

学習結果の要約と, 損失の変化が図 6 の左側に出力される.

学習後のネットを trainednet3 と定めて, 次のように入力しテストデータ (1,2,3 組) の分類測度を求める.

$$measurements = ClassifierMeasurements[trainednet3, testset[1,2,3]]$$

正確さは 1 であり分類測度と分類表が図 6 の右側に出力される.

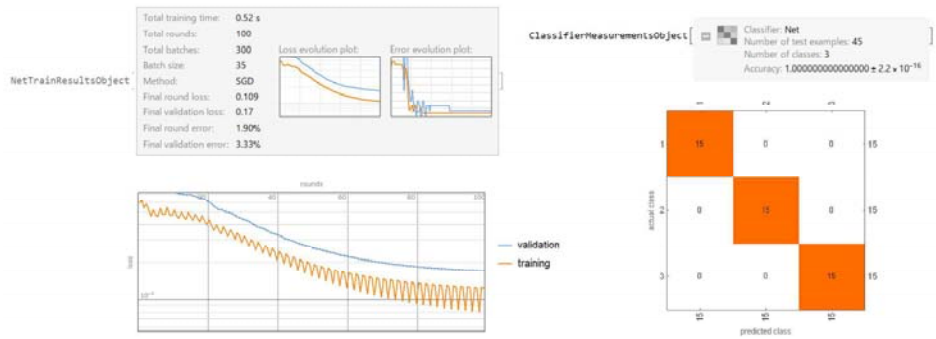

図 6: アイリスデータ (1,2,3 組) の学習, 分類

4 学習モデルにおける過学習の分析

4.1 定義

定義 11 (関数近似モデル, 学習モデル)

パラメータ $w \in \mathbb{R}^n, \mathbb{R}^1$ への関数 $f(x,w)$ に対して, 次で定まる $\mathbb{R}^1$ 上の確率変数 $Y = f(x,w) + Z$ を関数近似モデルという. ここで, 正規分布に従う確率変数 Z を雑音という. $y \in \mathbb{R}^1$, 分散 σ に対して, 関数近似モデル Y が従う条件付確率 $p(y|x,w) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{|y-f(x,w)|^2}{2\sigma^2}\right)$ を学習モデルという.

$-3 \leq x \leq 3$ の $\mathbb{R}^1$ 上の一様分布に従う確率変数 X を入力とする. 雑音 Z に対して, $Y = 5\tanh(3x) + Z$ で定まる $\mathbb{R}^1$ 上の確率変数 Y を出力とする. ここで, 雑音 Z が平均 0, 分散 $\sigma = 0.15$ の $\mathbb{R}^1$ 上の正規分布に従うとする.

定義 12 (真の分布)

出力 Y が従う条件付確率 $q(y|x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{|y-5\tanh(3x)|^2}{2\sigma^2}\right)$ を真の分布という.

定義 13 (2 乗誤差関数)

学習モデル $p(y|x,w)$ に対して, 2 乗誤差関数 $L_n(w) := -\frac{1}{n}\sum_{i=1}^n (y_i - f(x_i,w))^2$ で定める.

Mathmatica を用いてテストデータとニューラルネットワークの出力の 2 乗誤差を求めるため, $meanTestLoss[net] := SquaredEuclideanDistance[net[testX], testY]/Length[testX]$ と入力する [6].

定義 14 (RMS(二乗平均平方根) 重み)

重み $w_1, w_2, \cdots w_n$ に対して, RMS(二乗平均平方根) 重みを $\sqrt{\frac{1}{n}\sum_{i=1}^n {w_i}^2}$ で定める.

4.2 過学習を起こす場合

30 個の入力 xs, 出力 ys, 入力から出力への訓練データを Mathematica を用いて次のように作成する [7].

$$xs = RandomReal[UniformDistribution[-3, 3], 30];$$
$$ys = 5Tanh[3xs] + RandomVariate[NormalDistribution[0, .15], 30];$$
$$data = Normal[AssociationThread[xs- > ys]];$$

ここで入力を $dataX = Keys[data]$; 出力を $dataY = Values[data]$; として取り出すことができる.
30 個の入力 x, 出力 y, テストデータ $testData$ を xs, ys, $data$ と同様に作成する.
ここで入力を $testX = Keys[testData]$; 出力を $testY = Values[testData]$; として取り出すことができる.

4.2.1 ニューラルネットワークの作成, 訓練データの学習

学習モデルとして中間ユニット数を 1000 である 3 層ニューラルネットワーク（第 3 層の重みを 0, 第 3 層のバイアスをなしとする.）を作成するために次のように入力すると, 以下の図 7 の左側のように出力される.

$$net = NetChain[\{LinearLayer[1000], Tanh, LinearLayer[1, "Weights"- > \{Table[0, 1000]\}, "Biases"- > None]\}]$$

2 乗誤差を損失関数として, ニューラルネットワークに対して 50000 ラウンド学習させるために次のように入力すると, 学習結果の要約が以下の図 7 の右側のように出力される.

$$results1 = NetTrain[net, data, All, MaxTrainingRounds- > 50000]$$

図 7: ニューラルネットワーク, ネットワークの学習

学習後のニューラルネットワークを overfitNet と定めて, テストデータに対する 2 乗誤差を計算するために, $meanTestLoss[overfitNet]$ と入力すると 0.11079 と出力される. 訓練データに対する経験損失 0.00588716 より大きく, 過学習が起こったことが表示される.
訓練データとテストデータに対して, ニューラルネットワークの出力値を表すために次のように入力する.

$$Show[Plot[overfitNet[x], x, -3, 3], ListPlot[List@@@data, PlotStyle- > Red]]$$
$$Show[Plot[overfitNet[x], x, -3, 3], ListPlot[List@@@testData, PlotStyle- > Orange]]$$

訓練データ上への出力値は図 8 の左側に, テストデータ上への出力値は図 8 の右側に出力される. 学習後のニューラルネットワークはテストデータをうまく近似できない.

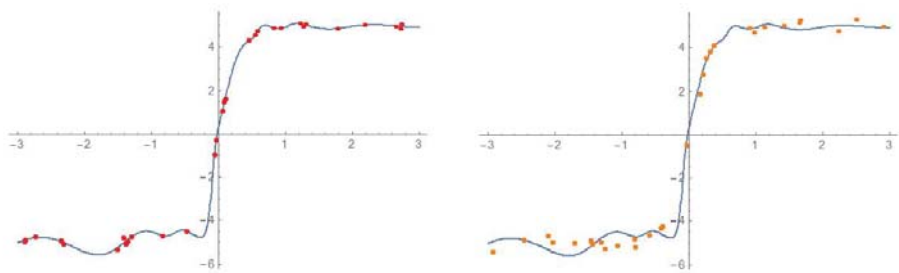

図 8: ニューラルネットワークの出力値 (過学習)

最大 10000 ラウンド学習する中でテストデータの損失が最小となるラウンド数で学習を終えるように，訓練データを学習させるために次のように入力する．

$$results2 = NetTrain[net, data, All, ValidationSet- > testData, MaxTrainingRounds- > 10000]$$

損失が最小であるラウンド数は 6438 であり，学習結果の要約と，損失の変化が図 9 に出力される．

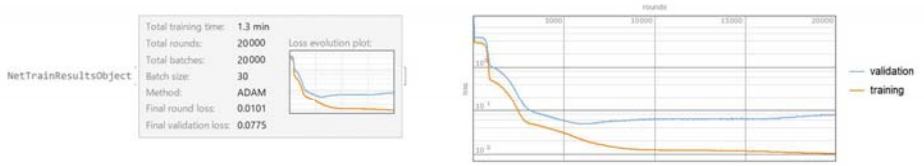

図 9: ニューラルネットワークの学習 (過学習)

よって早期に学習を終えたラウンドでの経験損失 0.0473004 は最終ラウンドでの経験損失 0.0775414 や，テストデータに対する 2 乗誤差 0.11079 よりも小さい値である．

早期に学習を終えたニューラルネットワークを earlyStoppingNet と定めて，訓練データとテストデータに対して，ニューラルネットワークの出力値を表すために次のように入力する．

$$Show[Plot[earlyStoppingNet[x], x, -3, 3], ListPlot[List@@@data, PlotStyle- > Red]]$$
$$Show[Plot[earlyStoppingNet[x], x, -3, 3], ListPlot[List@@@data, PlotStyle- > Purple]]$$

訓練データ上への出力値は図 10 の左側に，テストデータ上への出力値は図 10 の右側に出力される．学習後のニューラルネットワークはテストデータをうまく近似する．

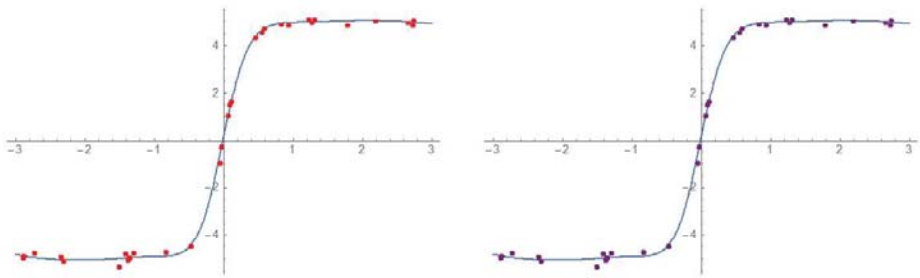

図 10: ニューラルネットワークの出力値のグラフ

4.3 過学習を起こさない場合

4.3.1 ニューラルネットワークの作成, 訓練データの学習

学習モデルとして中間ユニット数が 2 である 3 層ニューラルネットワーク（重みを第 1 層は 5 と 5, 第 3 層は 0 と 0, バイアスをなしとする．）を作成するために次のように入力すると，以下の図 11 の左側のように出力される．

$$net = NetChain[\{LinearLayer[2, "Weights"- > \{\{5\}, \{5\}\}, "Biases"- > None],$$
$$Tanh, LinearLayer[1, "Weights"- > \{\{0, 0\}\}, "Biases"- > None]\}]$$

2 乗誤差を損失関数として，ニューラルネットワークに対して 50000 ラウンド学習させるために次のように入力すると，学習結果の要約が図 11 の右側に出力される．

$$results1 = NetTrain[net, data, All, MaxTrainingRounds- > 50000]$$

図 11: ニューラルネットワーク, ネットワークの学習

学習後のニューラルネットワークを fitNet と定めて, テストデータに対する 2 乗誤差を計算するために, $meanTestLoss[fitNet]$ と入力すると 0.0202818 と出力される. 訓練データに対する経験損失 0.0184928 と比べて余り大きくなく, 過学習が起こらなかったことが表示される.

訓練データとテストデータに対して, ニューラルネットワークの出力値を表すために次のように入力する.

$$Show[Plot[fitNet[x], x, -3, 3], ListPlot[List@@@data, PlotStyle- > Red]]$$
$$Show[Plot[fitNet[x], x, -3, 3], ListPlot[List@@@testData, PlotStyle- > Orange]]$$

訓練データ上への出力値は図 12 の左側に, テストデータ上への出力値は図 12 の右側に出力される. 学習後のニューラルネットワークはテストデータを近似する.

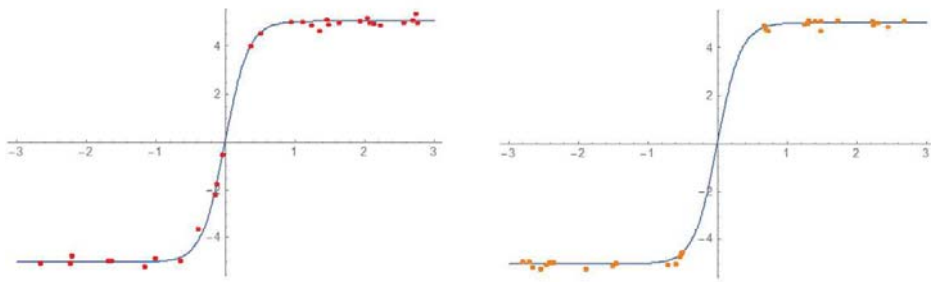

図 12: ニューラルネットワークの出力値

4.3.2 RMS(二乗平均平方根) 重みのグラフ

10000 ラウンド学習する過程で変化する RMS 重みをグラフに表すために次のように入力すると, 以下の図 13 の左側に出力される.

$$results2 = NetTrain[net, data, "RMSWeightsEvolutionPlot", MaxTrainingRounds- > 10000]$$

2000 ラウンド学習する過程で変化する RMS 重みの値を, 次のように入力して求める.

$$results3 := NetTrain[net, data, "RMSWeightsHistories", MaxTrainingRounds- > 2000]$$

学習前のニューラルネットワークを net1 と定めて, RMS 重みの初期値を, 次のように入力して求める.

$$N1 = RootMeanSquare[Flatten[NetExtract[net1, 1, "Weights"]]];$$
$$N2 = RootMeanSquare[Flatten[NetExtract[net1, 3, "Weights"]]];$$

スケールを変更せずに RMS 重みの変化をグラフに表すために次のように入力すると, 以下の図 13 の右側に出力される.

$$t1 = TimeSeries[Join[N1, results3[1, "Weights"]]];$$
$$t2 = TimeSeries[Join[N2, results3[3, "Weights"]]];$$
$$ListLinePlot[t1, t2, PlotRange- > All, PlotLabels- > 1, "Weights", 3, "Weights"]$$

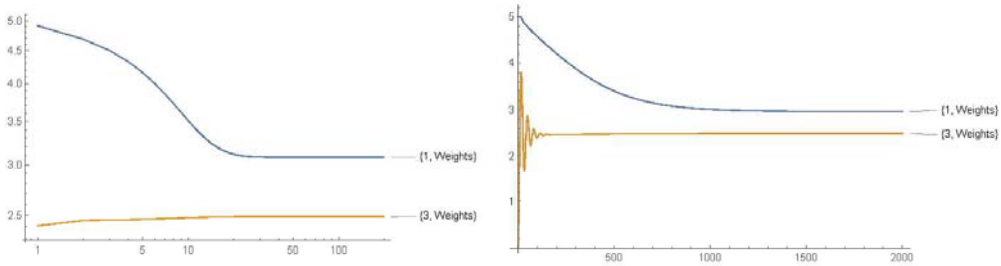

図 13: RMS 重みのグラフ

4.3.3 重みの収束値

10000 ラウンド学習させて最終ラウンドでの重みの値を求めるために次のように入力する.

$$result4 = NetTrain[net, data, "FinalWeights", MaxTrainingRounds- > 10000]$$

以下のように出力され, 第 1 層の重みは約 3 であることが表示される.

$$< |\{1, "Weights"\}- > \{\{2.98904\}, \{2.98904\}\}, \{3, "Weights"\}- > \{\{2.49893, 2.49893\}\}| >$$

第 1 層の重みを正とする. このとき第 3 層の重みの和を求めるために次のように入力する.

$$w3 := Flatten[results4[3, "Weights"]]$$
$$RealSign[First[results4[1, "Weights"]]] * First[w3] + RealSign[Last[results4[1, "Weights"]]] * Last[w3]$$

4.99974 と出力され, 第 3 層の重みの和は約 5 であることが表示される.

4.3.4 真の分布を実現するパラメータの集合

学習モデル $p(y|x, w)$ と真の分布 $q(y|x)$ が等しくなる代数的集合 V を考える.

$$\begin{aligned} V &:= \{w \in \mathbb{R}^4 | p(y|x, w) = q(y|x)\} = \{w \in \mathbb{R}^4 | a_1 \tanh(c_1 x) + a_2 \tanh(c_2 x) = 5 \tanh(3x)\} \\ &= \{w \in \mathbb{R}^4 | a_1 c_1 + a_2 c_2 - 5 \cdot 3 = a_1 {c_1}^3 + a_2 {c_2}^3 - 5 \cdot 3^3 = a_1 {c_1}^5 + a_2 {c_2}^5 - 5 \cdot 3^5 = 0\}. \end{aligned}$$

パラメータ表示は次のように表される [8].

$(a_1, 0, \pm 5, \pm 3)$, $(0, c_1, \pm 5, \pm 3)$, $(\pm 5, \pm 3, a_2, 0)$, $(\pm 5, \pm 3, 0, c_2)$, $(\pm 5\lambda, \pm 3, \pm 5(1-\lambda), \pm 3)$, $(\pm 5\lambda, \pm 3, \mp 5(1-\lambda), \mp 3)$.

最終ラウンドでの第 1 層と第 3 層の重みの値は代数的集合のパラメータ表示に含まれていることが分かる.

参 考 文 献

[1] 渡辺澄夫, 代数幾何学と学習理論. 森北出版株式会社, 2006.

[2] Sumio Watanabe, Algebraic Geometry and Statistical Learning Theory, Cambridge Univesity Press 2009.

[3] 渡辺澄夫, ベイズ統計の理論と方法. コロナ社, 2012.

[4] Sumio Watanabe, Mathematical Theory of Bayesian Statistics, CRC Press, 2018

[5] Mathematica チュートリアル, http://reference.wolfram.com/language/tutorial/ NeuralNetworksIntroduction.html

[6] Mathematica チュートリアル, http://reference.wolfram.com/language/tutorial/NeuralNetworksRegularization.html

[7] Mathematica チュートリアル, http://reference.wolfram.com/language/tutorial/NeuralNetworksRegressionWithUncertainty.html

[8] 鷲野朋広, 高橋正, 学習モデルにおける補題の証明, 数理解析研究所講究録 2138, 2019.