機械学習の初歩

１.1 はじめに

‘機械学習’を、よく、目にしたり、耳にするようになりました。‘機械学習’を、たしなみ程度で、知りたい人向けに、まとめました。

・数式は、極力避けようと思います。

・やさしい日本語で、記載するように、努めます。

楽な説明を目指しますので、興味あるところは、別途学んでください。

機械学習を、簡単に説明すると、（荒っぽい説明になります）

・回帰と分類があります。回帰は、明日の気温を予測することで、分類は、明日の天気を晴か曇りか雨か予想することです。両者の差は、最後の出力方法によります。

・数式を扱います。パラメータとバイアス値を求めますが、〇〇の法則のようなものはなく、ひたすら計算して求めます。計算機で行うのですが、この時間を短くするための手法が多くあります。（勾配の解析手法、誤差逆伝搬法　とかです）

1.2 ソフトについて

機械学習は、手計算すると長くなるので、機械にやってもらおうという考えですので、機械を動かすには、処理を打ち込んだコードが必要です。

（メニューを選んで処理する商用ソフトがありますが、ここでは言及しません。）

R　と　Python（いずれも無償です）が、分析ソフトで、有名です。

R の方がデータ分析に特化していると言われますが、Python 講習会の先生は、「Python を使用していて、困ったことはとりあえずない」との弁でした。この二つは、ライブラリが充実しています。（ライブラリとは、例えば、データ分析用に作成された、関数の集まりと理解してください）

C 言語も、著名なソフトですが、ライブラリが充実してないようです。

（自分で書けば、問題ないですし、コアな部分への取組は C 言語かもしれません）

Excel も、機械学習にどこまで対応できるは不明ですが、使いやすいです。

関数は豊富で、VBA でプログラムもでき、グラフも作りやすいです。

一番普及していて、ファイル 1 個にデータ、レポート、コードも混在可能です。

Web 記事が多いのは、Python、Excel で、困り事の検索には、優れます。

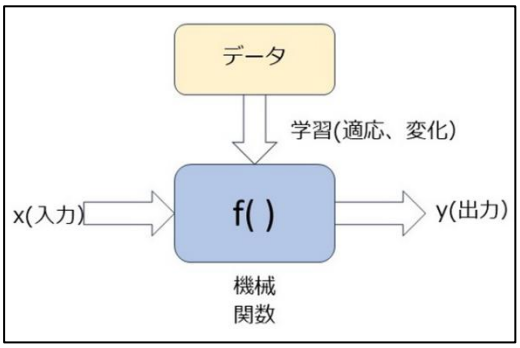
エディタのお試しであれば、Google Colaboratory が適当かと思います。

・Python が使えて、ライブラリのインポート済です。GPU にも対応してます。

・無料版ですと、12 時間の使用が上限のようです。（しばらくアクセスしないと注意されますが）性能、容量についても お試しであれば、問題ないと思います。

2 機械学習

機械は、右図に示すように、入力に対して、決まった出力を返してくれます。



機械は関数 f()にもとづいて、計算するので、学習用のデータを用いて、関数 f()を、適切に変更します。

y を目的変数、 x を説明変数として、説明されることも多いです。

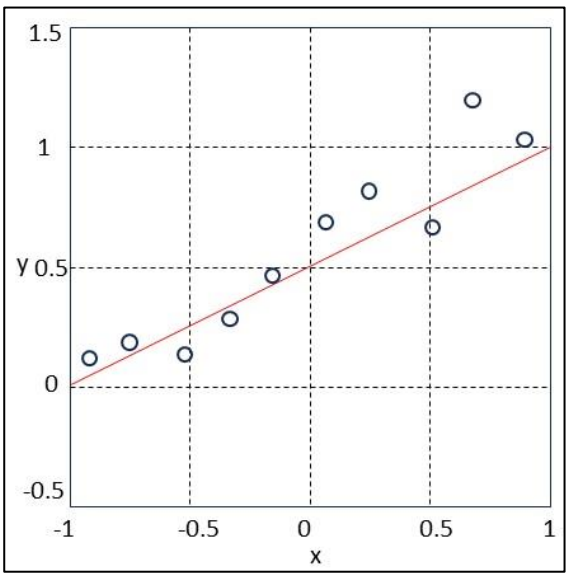
以上のことを「回帰」「分類」「深層学習」に分けて、説明します。

2.1 回帰

単回帰と重回帰について説明します。

2.1.1 単回帰

右図のように、〇印の観測値を、直線で回帰すると、 $\hat{y} = a + bx$ という数式になります。



y の頭にある ^ （ハット）は、予測値(期待値)の意味です。 y は実際の値です。

最小二乗法で、 a と b を決めます。

y と $\hat{y}$ の差の全データでの総和が、最小に

なる、a と　b　を求めます。

計算機でおこないますが、差を、a 及び、b　で微分して 0 になる値を求めます。

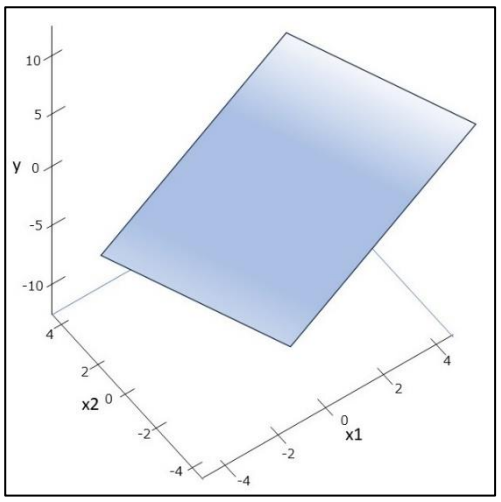
（この計算について、ライブラリで、関数が用意されています）

2.1.2 重回帰

x が 1 次元でなく、D次元の時、重回帰と呼ばれます。

$\hat{y} = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_D x_D$　の数式です。

D = 2 の時　右のような平面のグラフになります。（適当に書いた絵です。）



D が 3 以上になると、'超平面'という面になり、グラフでは表示できないようです。

重回帰においても、最小 2 乗法で、係数を、決定しますが、　y , w, x を、

ベクトルとして計算処理しています。

（この計算についても、ライブラリで、

関数が用意されています）

豆知識

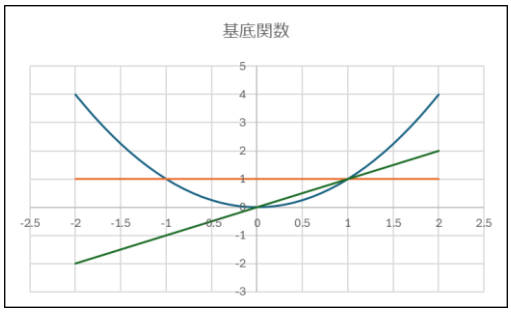
・曲線のモデルは可能か

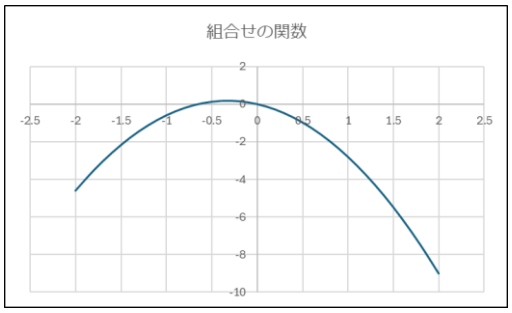
関数のモデルは直線（1 次式）としてますが、曲線のモデルは可能でしょうか

「線形回帰」を紹介します。「線形」とついていますが、これは関数のモデルが 1 次関数（直線）ということではなく、モデルの式が線形結合の形をしているということです。

次頁で、左の基底関数のグラフは、 $\Phi_0(x)=1$(橙色), $\Phi_1(x)=x$(緑色), $\Phi_2(x)=x^2$ (青色) を、プロットしています。右の組合せ関数のグラフは

$\hat{y} = 0.9\ \Phi_0(x) \ -2\ \Phi_1(x) \ -1.7\Phi_2(x)$　　をプロットしています。





基底関数は　三角関数、対数関数、指数関数 等も対応可能ですので、

曲線モデルも対応できそうです。ライブラリについては、探すことが出来てません。

・バラツキについて

　単回帰の項のグラフで、実測値(〇印) は、モデルの直線の上下にちらばります。

誤差 ε として、式に加えることがあります。y = a + bx + ε　であり、

誤差 ε　は、正規分布し、単体のグラフにすると、0 をセンターに上下に同等の値で、

プロットされると考えます。

・説明変数について

　重回帰では、説明変数が複数になり、扱いに注意を要します。

当たり前ですが、変数は少なくすることが第一歩です。目的変数に関係ないような変数は

削除したり、相関が見えるように対数変換したりします。

また、変数間で強い相関があってもよろしくないようです。(誤判定する)

　評価においても、予想値の良さで評価すると過学習の弊害が出るので、

相関係数で評価すべきとあります。

・最尤法と最小二乗法

　係数の決定を最小二乗法で行うとしましたが、最尤法というのもあります。

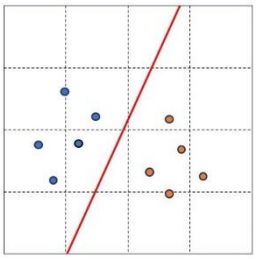
ただ、線形回帰において、誤差が正規分布にしたがうことを仮定すると、

最小二乗法と最尤法は、同じ結果になります。

　最尤法を、少し紹介すると、確率関数を用いて、説明変数と目的変数の関数関係を求め、尤度が、最大になるような関数を求めます。作法は、最小二乗法と異なりますが、線形、正規分布の前提では、結果は同じです。

2.2　分類

y を目的変数と紹介しましたが、y が量的な変量であれば、回帰であり、質的な変量であれば、分類といえます。　例えば、y = f(x) において、y が、'yes' または 'no' という出力になるという意味です。



右図は、簡単な例ですが、青丸と橙丸が、赤線で、分類されている図です。

代表的な、手法を紹介します。

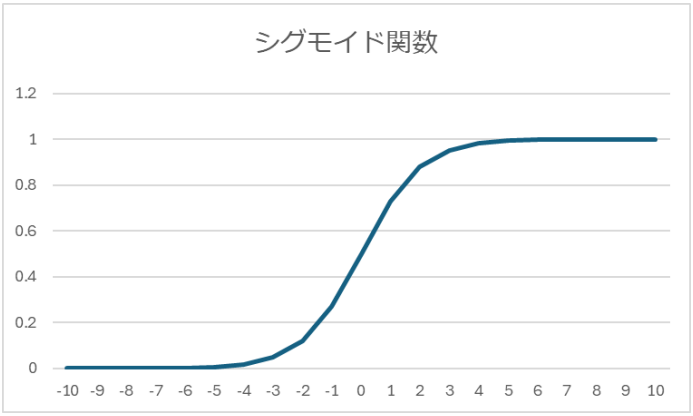
2.2.1　ロジスティック回帰分析

2.1.2 項の重回帰で用いた数式で、分類を行います。

$z = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_D x_D$　(重回帰の式)

$y = \frac{1}{1+\exp(-z)}$　(シグモイド関数と呼ばれ、下がプロットです)

y は　0 から 1 の値を取るので、確率を出力していて、二値分類を行っています。

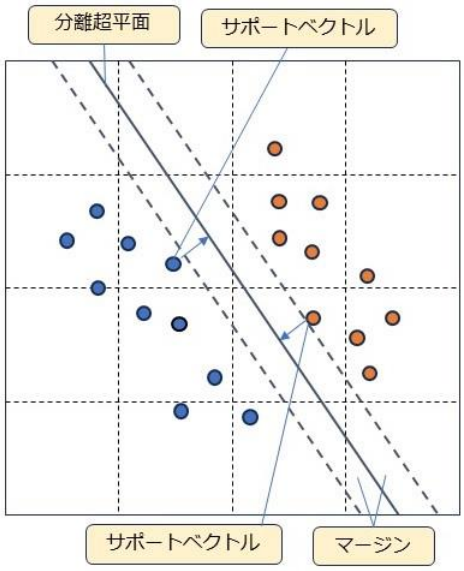


2.2.2　サポートベクターマシン

データを分類して境界線を引くための手法です。実測値をもとに境界線を定めたを作成し、未知のデータを入力して、分類を行います。

・マージンの最大化

データ点が境界線から最も近い位置にある点をサポートベクトルと呼び、このサポートベクトルと境界線の距離（マージン）が最大になるような境界見つけ、「分離超平面」とします。



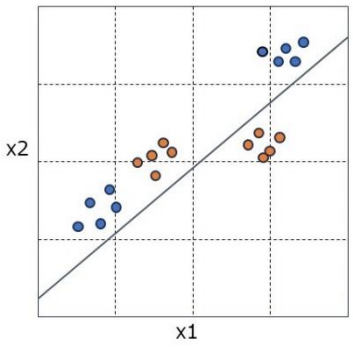
ハードマージンとソフトマージンとが
あり、右の例は、ハードマージンですが、
実際は、このように、ハッキリと分離できない
ので（外れ値等発生します）、ある程度の
誤分類も許容するのが現実的です。
この場合を、ソフトマージンと言います。

分離超平面は、

$w_1\ x_1\ +\ w_2\ x_2\ +\ \cdots\ w_n\ x_n\ +\ \mathrm{b}\ =\ 0$ となり、

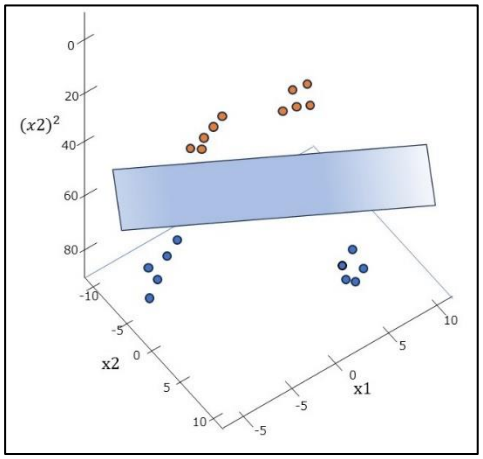
最大マージンとなる、係数 w を求めます。分離超平面が定義できたら、未知のデータを入力して、 0 より大か小かで、分離します。

・カーネル法



右図の様に、直線では分断できないことがあります。
そういう時の対応で、カーネル法があります。
2 次元（平面）データを 3 次元などの高次元空間へと
拡張し、平面でしか表現できなかったデータを立体的に
表現することで「平面による分離」を追加する方法です。

図を例にすると、(x1,x2) の二次元ですが、(x1,x2,x^2)の 3 次元にします。（写像というそうです）



3 次元に写像したプロットが右図です。
「分離超平面」で分離しています。
（計算したわけではなく、イメージ図です）

豆知識

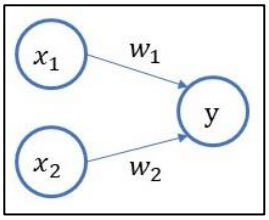
分類について、二つの手法を紹介しました。
ライブラリが用意されています。その他の手法と
して、決定木、ディープラーニング、
ニューラルネットワークが、あります。

2.3　深層学習

深層学習は、回帰、分類のように 変数の関係を手掛かりにするものではないので、 少し毛色が違うように感じます。

ここでは、深層ニューラルネットワーク(DNN=Deep Neural Network) と 畳み込みニューラルネットワーク（CNN=Convolutional Neural Network ）と 再帰型ニューラルネットワーク（RNN =Recurrent Neural Network）について説明します。

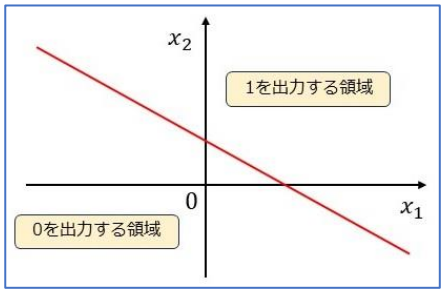
2.3.1　深層ニューラルネットワーク(DNN)



右の図は、深層学習のベースとなる考えです。
深層学習は、更に複雑で手の込んだものですが、簡単なところで、説明します。右図で、x_1 と x_2 が入力されます。
w_1 と w_2 はパラメータで、重みとも呼びます。

Z = w_1 x_1 + w_2 x_2 + b （ b はバイアス、切片と呼ばれるものです)

Z ＞0 の時は y = 1 , Z <0 の時は y = 0 を出力させるとします。

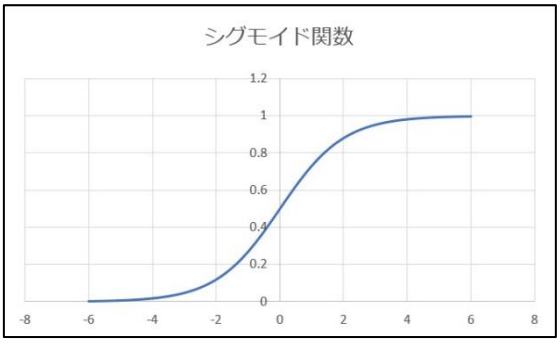


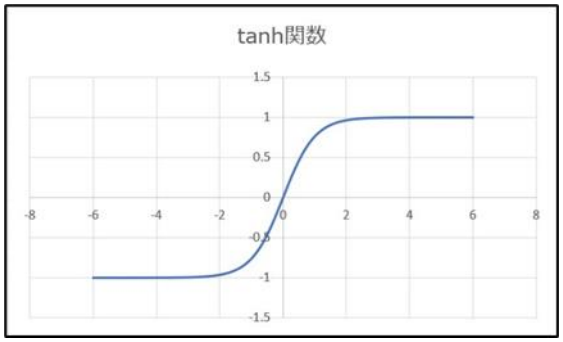
右図において 赤線は、Z = 0 の直線です。
重み、バイアスを調整することで、最適な直線を得ることができます。

「Z> 0 :y = 1 , Z<0: y = 0」の出力 は、活性化関数で、求めます。

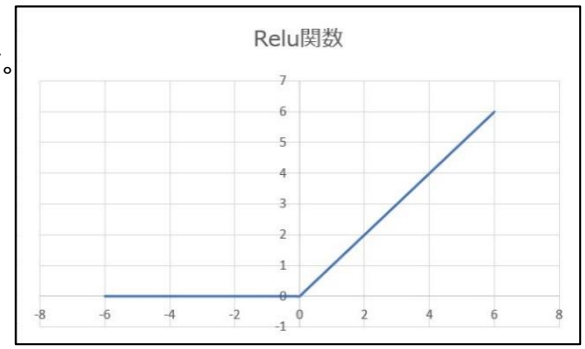
活性化関数は、シグモイド関数（h = $\frac{1}{1+\exp(-z)}$ ） （左側）とか

ハイパボリックタンジェント関数 (h = tanh(x) ）(右側) があります。





加えて、ReLU 関数も紹介します。
パラメーターの最適化が高速で行えます。
シグモイド関数とハイパボリックタンジェント関数は、x の増加により、傾きが 0 になるので、勾配ベクトルの大きさが小さくなり、最適化処理が進みにくくなるようです。



3 層だけのニューラルネットワークを紹介します。入力層が二つ、中間層が三つ、出力層は、二つです。今までの延長線上にありますが、行列の計算があるので、興味ない方は、飛ばしてください。（行列の計算は深層学習につきものです。）

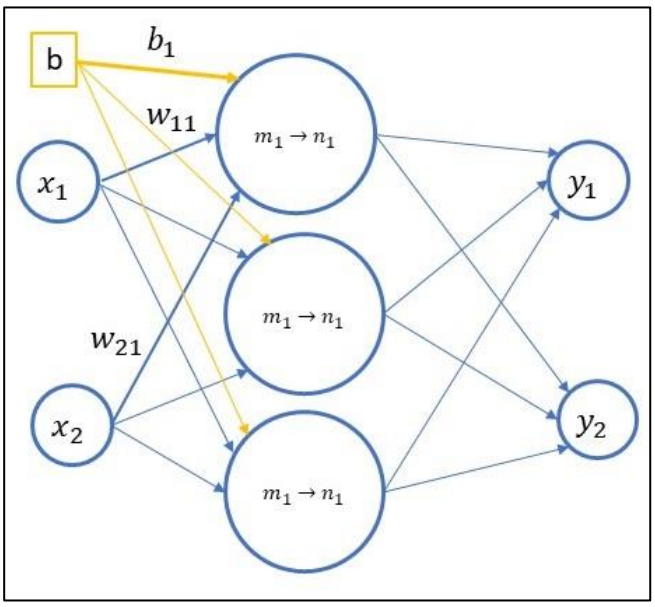
入力層と出力層は、事象から決まりますが、中間層は、任意で、設計する必要があります。（ハイパーパラメータと呼ばれます）

また、このように、層間が密に結合されるものを、全結合型と言います。

右図で、 b はバイアスです。
w はパラメータ（重み）です。
wの右下の数字は、1 桁目が 1 層目の番号、2 桁目が、2 層目の番号を示します。

n_1 は、m_1 を活性化関数で変換したものです。

上述の一層を例に習って、式を構成します。



$$m_1 = w_{11}\ x_1 + w_{21}\ x_2 + b_1 \qquad n_1 = \mathrm{h}(\ m_1\ )$$

$$m_1 = w_{11}\ x_1 + w_{21}\ x_2 + b_1 \qquad n_1 = \mathrm{h}(\ m_1\ )$$

$$m_1 = w_{11}\ x_1 + w_{21}\ x_2 + b_1 \qquad n_1 = \mathrm{h}(\ m_1\ )$$

これを行列でまとめます。

$$(m_1 \quad m_2 \quad m_3) = (x_1 \quad x_2)\begin{pmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{pmatrix} + (b_{11} \quad b_{12} \quad b_{13})$$

$(n_1 \quad n_2 \quad n_3) = \mathrm{h}\left((m_1 \quad m_2 \quad m_3)\right)$ となります。

さらに、3 層目の出力層は、以下となります。

$$(y_1 \quad y_2) = (n_1 \quad n_2 \quad n_3)\begin{pmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \end{pmatrix} + (b_{21} \quad b_{22})$$

例えば y_1= 0.15 , y_2= 0.85 となるとy_2に対応するラベルに分類されます。

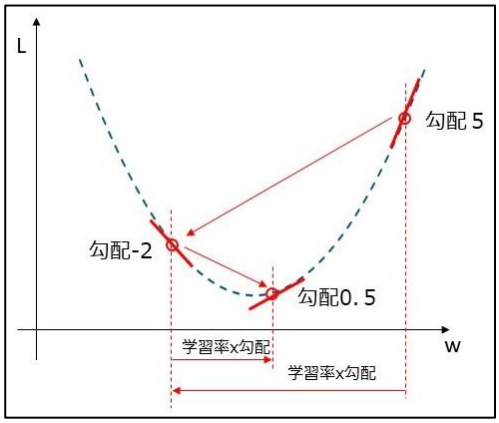
次に、パラメータをどのように求めるかについて、説明します。

計算機を用いて繰り返し数値計算を行います。数値的な手法で最適なパラメータを求めることであり、数値解と呼ばれます。

つまり、初期値を事前に決めて、初期値を更新していく方法になります。

最適なパラメータの指標を表す関数を損失関数と呼び、平均二乗誤差（回帰で使用）交差エントロピー（分類で使用）が代表的で、値として、最小の時に、最適なパラメータとなる数式となっています。（詳細は数式の連続で省略します）

最小値の求め方は、勾配降下法があり、バッチに分けて勾配降下法を用いるミニバッチ学習があります。右図は、y 軸が損失関数(L) x 軸がパラメータ(w)です。二次関数として、L の最小値を求める方法です。勾配 5 の点が初期値です。勾配がプラスですので、最小を求めて左手に移動させます。



一般に、（学習率） x 勾配分）とされています。移動後の位置が勾配-2 の点です。さらに、右手に移動します。この繰り返しで勾配 0 を探します。

ミニバッチ学習というのは、全データを、いくつかのデータの分けてまとめて、それぞれの勾配を計算します。その勾配の平均値を用いて、パラメータを更新します。

このような方法を、確率的勾配降下法(SGD) と呼び、ニューラルネットワークの主流となっているよでうです。(計算時間が短かくできる)

最後に、分類で用いるソフトマックス関数を紹介します。式では以下です。

$$y_k = \frac{\exp(a_k)}{\sum_{i=1}^{n} \exp(a_i)}$$

出力層が n 個あって、k 番目の出力値を全出力値合計を分母にして求めています。

これは、複数の入力した値を、出力 0 から 0.1　で総和を 1 にする関数です。

出力を確率とみなせるので、分類によく使用します。

豆知識

誤差逆転伝播法について

ニューラルネットワークにおいて、パラメータ（重み）とバイアスを訓練データに適応するように対応するのが学習です。手順は以下です。

・訓練データから、損失関数の勾配を求めます。(勾配の算出)

・パラメータを勾配方向へ更新します。(更新)→　この繰り返しで勾配 0 を求めます。

この手順を、ネットワークの入力側の左側からの順方向から計算するか、出力側の右側からの逆方向から計算する（誤差逆転伝播法）かがあります。

計算量が少ない誤差逆転伝播法（微分を合成関数で適用するので計算が少ないです。）

で行うのが主流のようです。

確認のため、順方向から計算で、勾配確認することを行います。

ライブラリとして、forward , backward の関数が用意されています。

活性化関数

0 から 1 へステップ状に変化するステップ関数がありますが、深層学習が、誤差逆転伝播法を活用するので、ステップ関数は勾配が発生しないため、不向きのようです。

シグモイド関数の使用もありますが、勾配が継続される ReLU 関数が主流とのことです。

2.3.2 畳み込みニューラルネットワーク（CNN ）

文字認識、画像認識が、よく聞きます。音声、ゲームの盤面の認識にも使われます。
私は、課題解決の体験はなく、文字認識と画像認識について、勉強用のデータセットで学んだ経験があるだけです。2 種類のデータセットを紹介します。

MNIST データセット

手書き数字画像のデータセットです。 6 万枚の訓練データ用の画像とラベル、1 万枚のテストデータ用の画像とラベルから構成されます。画像データに、0～9 のいずれかの手書き数字です。28×28 ピクセルのグレースケールの画像で、各ピクセルは 8 bits の単一の値です。（ライブラリにインストールされてます）

CIFAR-10 データセット

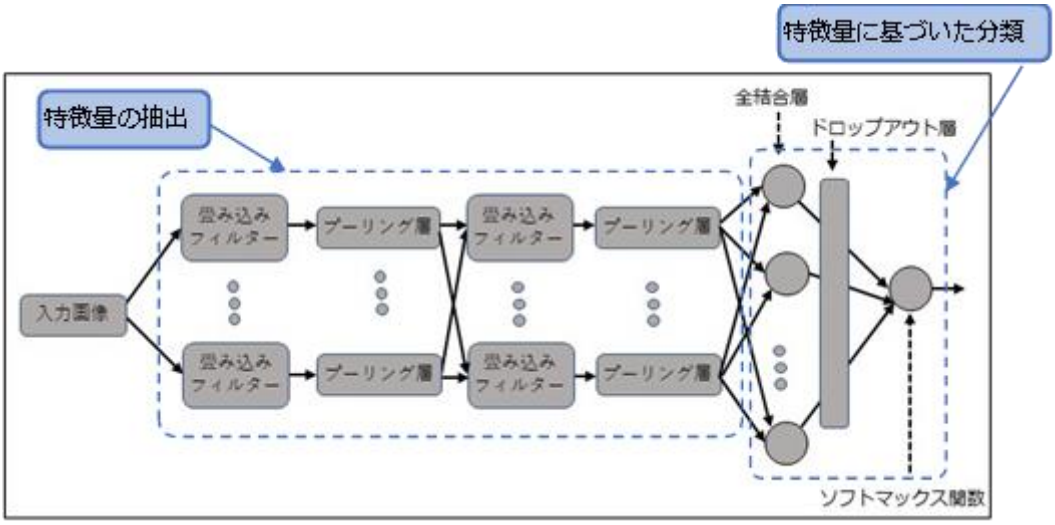
airplane（飛行機）automobile（自動車） bird（鳥）cat（猫）deer（鹿）dog（犬）frog（カエル）horse（馬） ship（船）truck（トラック）という 10 種類の「物体カラー写真」（乗り物や動物など）の画像データセットで、10 クラスの分類を行います。

5 万枚の訓練データ用の画像とラベル、1 万枚のテストデータ用の画像とラベルから構成されます。24bit RGB（赤色／緑色／青色）フルカラー画像で、「0」～「255」の 256 段階、画像は、幅 32×高さ 32 ピクセルで、 1 つ分のデータが基本的に(3, 32, 32)もしくは(32, 32, 3)（＝計 3072 要素）という多次元配列の形状となっています。
（ライブラリにインストールされてます）

注目すべきは万単位の訓練データが用意されているようで、現実の事象で使用するとしたら、分類の難易度・ラベル数にもよりますが、千枚単位の画像が必要なのかと思います。
(ここが一番知りたいところかと思いますが、開示されているのを見つけられませんでした。すいません)

概略を説明します。下図は CNN の構成図で、よく出てきます。

‘特徴量の抽出’と、‘特徴量に基づいた分類’ に分けられ、後者は、2.3.1 項と同じですので、ここでは省略します。学習データで、モデルを作成してテストデータで検証します。



‘特徴量の抽出’ の 畳み込み層と、プーリング層について説明します。

畳み込み層

入力されて画像にフィルター(カーネル)をかけて、特徴を分かりやすくする手法です。ここでは、28x28 のピクセルに、縦、横のエッジを抽出するフィルタを持ちます。

1 例として、左に画像の各ピクセルの色濃度の配列しました。 中央が、縦エッジを抽出するフィルタ（カーネル）で、右が計算の結果です。

3	0	2	2
1	0	3	1
3	2	2	0
1	2	2	1

-1	0	1
-2	0	2
-1	0	1

a=2	c=2
b=1	d=4

a = 3x-1 + 0x0 + 2x1 + 1x-2 + 0x0 + 3x2 + 3x-1 + 2x0 + 2x1 =2

b = 1x-1 + 0x0 + 3x1 + 3x-2 + 2x0 + 2x2 + 1x-1 + 2x0 + 2x1 =1

c = 0x-1 + 2x0 + 2x1 + 0x-2 + 3x0 + 1x2 + 2x-1 + 2x0 + 0x1 =2

d = 0x-1 + 3x0 + 1x1 + 2x-2 + 2x0 + 0x2 + 2x-1 + 2x0 + 1x1 =-4 (絶対値)

このフィルタは、横に伸びたエッジを、プラスとマイナスでキャンセルするので、縦に伸びたエッジを抽出します。（この例では顕著ではないですが）また、4x4 の枠が、2x2 になるので、あらかじめ周囲に、 0 の枠を追加して、フィルタリングすることも行います。（padding と言います）

プーリング層

特徴を明確にするため（白いか黒いかが分かれば良い）ので、解像度を落として、詳細情報を消します。具体的には、28x28 ピクセルの画像を、2x2 ピクセルをブロックにして、14x14 のピクセル画像に、

変換します。ピクセル値は、ブロック内の最大値を採用します。

3	0	2	2
1	0	3	1
3	2	2	0
1	2	2	1

3	3
3	2

次に、紹介した二つのデータセットにいつて、内容を紹介します。

（プログラムのコードは、長くなるので、別途とします）

MNIST で用いた層

・入力する画像は　28 x 28 サイズです。

・畳み込み層として、 5 x 5 サイズのフィルタを 16 個用い、パディングを'same' にして、同じ 28x28 のサイズにしてます。

・プーリング層で 14x14 サイズの 16 個の画像 にします。

・16 個の画像データで分類処理します。

・総パラメータは、3,222,954 個あり、テストデータで正解率 98 パーセントでした。

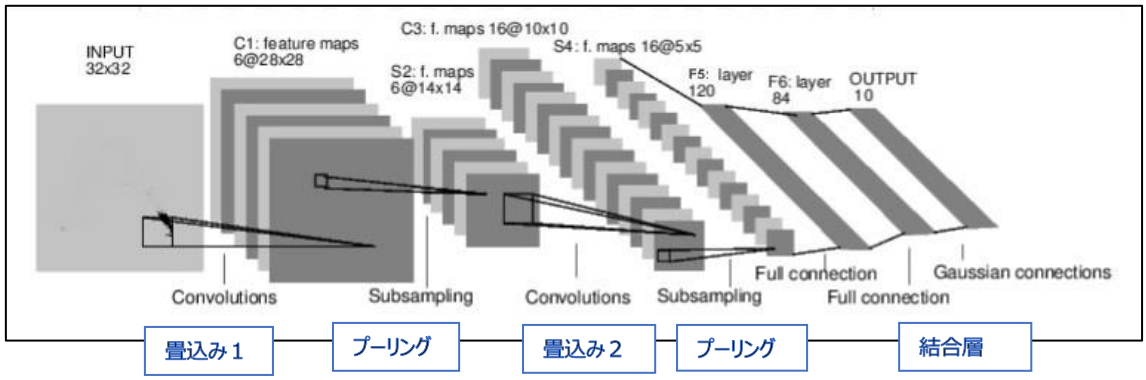
CIFAR-10 の層

今回の CNN のベースは 1998 年（古い！）に LeNet という名称で開発されました。

・入力する画像は　32 x 32

・畳み込み層 1 は、RGB の 3 チャンネルで、5x5 のフィルターを 6 個使用して、28x28 のサイズにしてます。

・プーリング層で、14x14 のサイズにしています。



　Python で提供しているライブラリとして TensorFlor2.0　と PyTorch　があります。処理速度は同等で、シェアとして優勢なのは、研究者は PyTorch、ビジネスユースまで拡大すると、TensorFlor2.0 になるようです。（2022 年度で）

豆知識

コンピューター上で扱える画像の形式

ベクター画像形式、ラスター画像形式の 2 種類があります。ベクター画像は、線や曲線といった図形集合の重ね合わせをデータとして保持した形式であり、ラスター画像はピクセル（画素ともいう）と呼ばれる色の数値を格子状に並べたデータを保持した形式です。。機械学習の画像はラスター画像形式のことになります。

フィルタ（フィルタカーネル）について

フィルタは、「3×3」「5×5」「7×7」などと、中心を決められる奇数が使いやすいです。

フィルタ数は、「16・32・64・128・256・512 枚」などが使われる傾向にあるようです

複雑そうな問題ならフィルタ数を多めに、簡単そうな問題ならフィルタ数を少なめで試してみるようです。例として、エッジを強調するフィルタ（カーネル）、平均化するフィルタ（カーネル）を紹介します。

0	1	0
0	-2	0
0	1	0

縦方向エッジ強調

0	0	0
1	-2	1
0	0	0

横方向エッジ強調

0	1	0
1	-4	1
0	1	0

縦横方向エッジ強調

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

平均化フィルタ

GPU について

深層学習は、大量の演算を行うので、高速化の取組が重要です。

その一つが、GPU(Graphics Processing Unit)の採用です。

GPU は、グラフィックのための専用デバイスで、並列的な数値計算を可能とします。

深層学習は、大きな行列の積計算を必要とするため、GPU が有効なようです。

特に、2006 年に、NVIDIA 社から CUDA というプラットフォームが開発され、プログラムの GPU 対応が進化したとのことです。

2.3.3 再帰型ニューラルネットワーク（RNN）

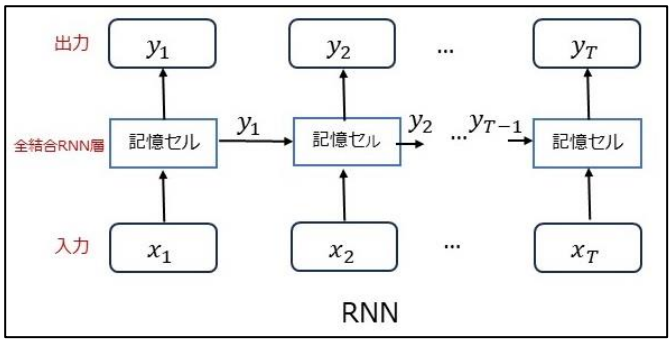
RNN は、順番のあるデータを扱うのが特徴です。時系列データ、 文章（文字や単語が順番に並んでいる）のデータの処理方法として、知られています。

ただ、RNN では、長期の系列については、勾配消失、勾配発散という問題があるので、LSTM(Long Short-Term Memory) という中間層に工夫をこらした方法が主流です。

図で、紹介します。

左図は、基本である RNN のネットワークです。x_t は入力で、1 ～ T まで続きます。

（時系列データでもあり、文章であれば、単語の羅列です）



記憶セルは、一般にシグモイド関数であり、これも 1 ～ T 番まで続きます。

よって、y_T は、x_T を全結合処理したものと y_{T-1} を全結合処理したものの和に、活性化係数を適用したものになります。このモデルも、訓練データを用いて最適なパラメータ（重み）を算出していきます。

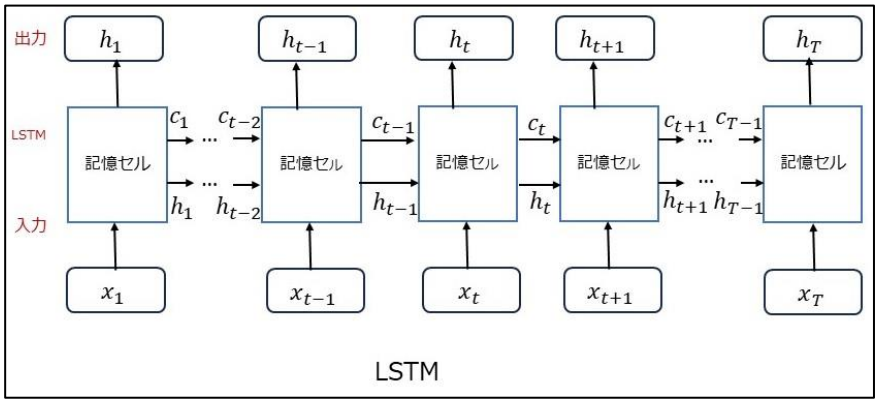
RNN は、長い系列には不向きとされています。理由として、上図の y は、シグモイド関数からの値ですので、 0～1 となります。逆転伝播法で、勾配を求めてると、微分の積であるので、 1以下の積算が続くと、勾配が 0 になるためとのことです。

上記の欠点の補完するのが、LSTM です。

右のようなネットワークになります。

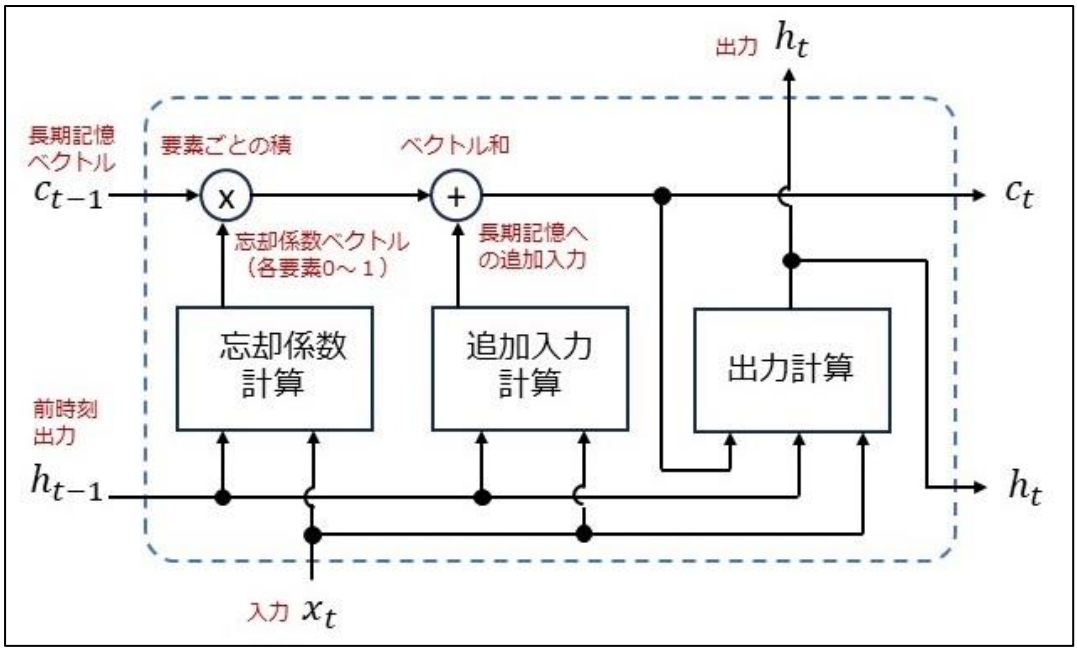
RNN との差異は、記憶セルの構造によります。

記憶セルについて説明します。



記憶セルは、以下の図ですが、簡潔に、数式無で説明します。

また、Tensflow.Keras　で、LSTM 関数が用意されてるので、処理できます。



長期記憶ベクトルを、随時更新（忘却して追加する）するのが、改善点のようです。

・忘却係数計算：現時刻の入力と前時刻の出力で、（重みで可変します）
シグモイド関数（0～１）を適用させて、前段階の長期記憶ベクトルに積算して、
一部を忘却させます。

・追加入力計算：活性化関数を tanh 関数（-1 ～1）とした追加情報計算と、シグモイド関数（0～1）とした係数計算との二つの層から構成され、現時刻の入力と前時刻の出力を、（重みで可変します）入力し積算した後、長期記憶ベクトルに追加します。

・出力計算：活性化関数を tanh 関数（-1 ～1）とした長期記憶ベクトルの計算と、
シグモイド関数（0 ～1）とした現時刻の入力と前時刻の出力による係数計算との二つの層から、構成され、長期記憶ベクトルに係数を積算して出力とします。

説明は以上です。

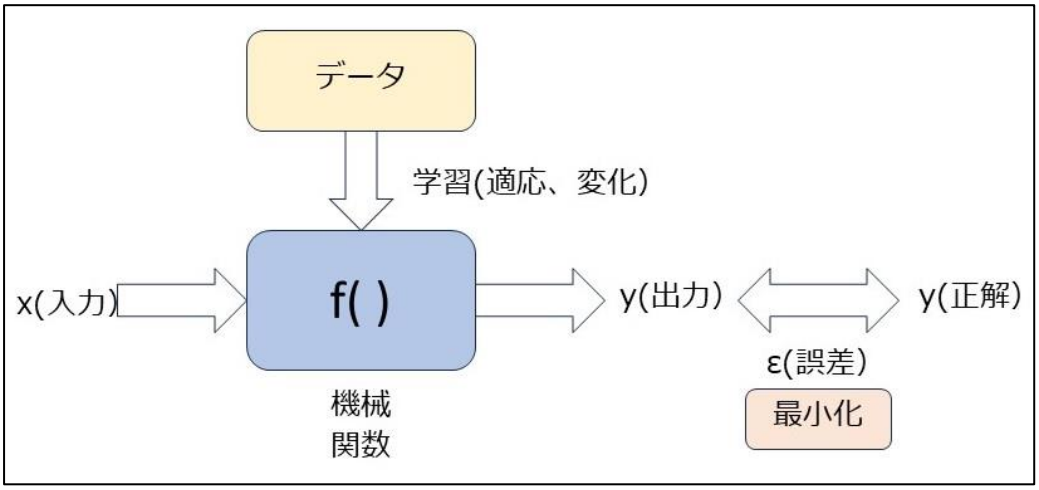
放電加工機の異常検知で適用したことがあります。問題なく異常検知に反応しました。

（エンコーダ・デコーダモデルの手法も必要です。）

3. 最後に

ここまで、読んでいただいて有難うございます。

最後に、繰り返しになりますが、機械学習を、簡単に説明します。



上図で

・データは訓練用とか学習用と言われるデータです。あらかじめ採取します。

（適切な、処理しやすいデータにするのがポイントです）

・f( ) は、予測モデルというもので、人が決めるところです。1 次式にするとか、層の数とかで、ハイパーパラメータと呼ばれます。

・予測モデルのパラメータを決めれば、予測モデルは完了です。

パラメータを w として、w を引数として関数を、損失関数 L(w) とします。

・w を動かして、y(出力)をもとめます。y(正解)との誤差を算出して、

最小となる誤差を示す、L(w) を探します。

上記ことを、手計算では大変です。2 次以上の関数であれば、超平面になります。

ここは、ライブラリが用意されているので、計算してもらおうということです。

以上ですが、機械学習について、このレポートでどこまで理解できたか不明ですが、

不明な点は、web で検索ください。

主なライブラリ（python ）

Numpy, pandas , matplotlib は、機械学習だけではなく、データ解析の基本のライブラリです。

Scikit-learn が機械学習、 TensorFlow Keras, Pytorch が深層学習用のライブラリの代表と言えます。これ以外のも多くあると思います。

参考にした文献と WEB

・ガウス過程と機械学習　持橋大地　大羽成征　講談社

・図解　深層学習　数理で理解する基本原理　　小池敦　近代科学社

・環境と品質のためのデータサイエンス　http://data-science.tokyo/index.html

・ニューラルネットワークの起源：パーセプトロンを図解と数式、プログラムで徹底解説
https://zero2one.jp/learningblog/origin-of-neural-networks/