第一章 機械学習と深層学習の導入

近年、機械学習や深層学習は毎日のようにネットやテレビ等でも取り上げられており、お茶の間レベルで聞かない日はないほどの盛り上がりを見せています。AIという名称でバズワード化されており、現在最も注目度が高いIT技術のひとつでしょう。本章では、機械学習を実践で使う際に必要となる基本的な知識を紹介します。すでに機械学習に関して十分知識をお持ちの方は、本章は読み飛ばしていただいて大丈夫です。それでは早速始めましょう。

1.1 従来のプログラミングと機械学習との違い

機械学習をはじめて触れた方は従来のプログラミングとの違いに戸惑う方がいらっしゃいます。そのためここでは、従来のプログラミングと機械学習の違いを説明します。Courseraという非常に有名なオンラインのAI系の学習サイトの情報に非常に理解しやすいものがあるので、そちらをもとにそれぞれの違いについて説明します。

従来のシステムでは以下の図のようにまず何かの課題に対してひたすらルールを自分でプログラミングします。そして、そのプログラムに対してデータを入れると答えが出てくるという、プログラム視点で見るとある意味受動的な構成になっています。

!5
https://www.coursera.org/learn/introduction-tensorï¬ow
 

従来のプログラミングの処理の流れ

たとえば人の活動の検知をしたい場合を考えます。

歩いているという状態をプログラミングする場合にどうすればよいでしょうか。以下の例は非常に概念的ではありますが、例えば動くスピードが4km/h以下であったら歩いているという風に定義をする方法があります。この方法で走っている状態を定義する場合には、else文を加えて4km/h以上だったら走っているというように定義できます。このようなやり方でどんどん機能拡張をしていくことを考えると、自転車に乗っている場合には、さらにelse文を加えてスピードが12km/h以上だったら自転車に乗っている状態と定義できます。このようにだんだんとやりたいことを増やしていくと、たとえばゴルフをしている状態なども実装したくなるかもしれませんが、どう言う風にそれを実装していいのか複雑すぎてお手上げになってしまいます。このような、人が簡単なロジックで記述することが難しい複雑なタスクに対しては機械学習が用いられるようになってきています。

Activity Recognition
!9
https://www.coursera.org/learn/introduction-tensorï¬ow
 

人の動作に関する従来のプログラミング例

従来のプログラミングと機械学習の違いを図にすると以下のようになります。

!10
https://www.coursera.org/learn/introduction-tensorï¬ow
 

従来のプログラミングと機械学習の処理の流れの違い

従来のプログラミングでは、ルールを決めてデータを入れてあげると答えが出てくるのでしたが、機械学習ではデータと答えのペアを与えてあげると、アルゴリズムがルールを作ってくれます。ここでのアルゴリズムは統計に基づいて確率的に機械学習モデルのパラメーターを調整します。ある程度説明力を持った手法も存在しますが、機械学習がよくブラックボックスだといわれるゆえんはたぶんこのようなところにあります。

たとえば下図のように、歩いている場面を撮った画像に、ラベルとして”歩いています”という属性をタグ付けします。同様に、走っている場面の画像には”走っています”、自転車に乗っている場面の画像には”自転車に乗っています”、ゴルフをしている場面の画像には“ゴルフをしています”というように画像データと答えにあたるラベルのペアをたくさん用意してモデルに与えてあげると、アルゴリズムがどういう風に推論をすればよいかを教えてくれます。

Activity Recognition
!11
https://www.coursera.org/learn/introduction-tensorï¬ow
 

機械学習による人の動作のプログラミングイメージ

1.2 機械学習のさまざまなアプローチ

機械学習では使用されるデータセットによる違いや答えの求め方の違いで大きく４つの学習方法に分類されます。用いる学習方法の違いによりアルゴリズムやモデルが異なるため、ここではそれぞれの学習方法の違いを理解しましょう。

| æ©æ¢°å­¦ç¿ã®æ§ããªã¢ãã­ã¼ã
â¢ æå¸«ããå­¦ç¿
â¢ æå¸«ãªãå­¦ç¿
â¢ åæå¸«ããå­¦ç¿
â¢ å¼·åå­¦ç¿
6
 

機械学習の学習方法の違いを大別すると上の図のように、教師あり、教師なし、半教師あり、強化学習の４種類があります。

それぞれの学習は大まかに以下のようになっています。

・教師あり学習： 入力と出力の関係を学習するアルゴリズム

・教師なし学習： データの構造を学習するアルゴリズム

・半教師あり学習： 教師あり学習と教師なし学習を組み合わせたもの

・強化学習： 報酬を最大にするような行動を学習するアルゴリズム

最初に以下のピザとドーナッツの画像の例を用いて、教師あり学習、教師なし学習、半教師あり学習ではどのようなことをやっているかについて説明します。

| ãã¶ã¨ãã¼ããã®åé¡
7
 

機械学習で用いる画像の例

教師あり学習とは

まず教師あり学習と教師なし学習では用いられるデータの形式に違いがあります。

教師あり学習では、以下の図のようにピザとドーナッツのそれぞれの画像に答えとなるピザやドーナッツのラベルがペアで用いられます。アルゴリズムに対して正解のラベルを与えることは、先生が生徒に対して正解を教えながら学習を進めていくことに似ているため、教師あり学習と呼ばれています。

それに対して、教師なし学習で用いられるデータには、前のピザとドーナッツの画像のように、それぞれの画像に対して正解にあたるピザかドーナッツかのラベルがついていません。あるのは単なる画像データだけです。

ãã¶
ãã¶
ãã¶
ãã¶
ãã¼ãã
ãã¼ãã
ãã¼ãã
ãã¼ãã
| æå¸«ããå­¦ç¿
8
 

教師あり学習に使う画像の例

教師あり学習が具体的にやっていることは、以下の図のように、ピザとドーナッツの間の決定境界と呼ばれる、できるだけ正しくピザかドーナッツかを見分けられる線を見つけることです。たくさんの画像から最適な境界線を引くルールを学んでいきます。

ãã¶
ãã¶
ãã¶
ãã¶
ãã¼ãã
ãã¼ãã
ãã¼ãã
ãã¼ãã
9
| æå¸«ããå­¦ç¿
æ±ºå®å¢ç
 

教師あり学習の分類のイメージ

教師なし学習とは

教師なし学習には、正解のラベルがついていないため、教師あり学習のように正解から学習することができません。しかし、本来ピザとドーナッツのことを全く知らない人間でも、少なくともなんとなくその形状から「これらは違うものだ」ということは想像できるでしょう。教師なし学習では、こういったことを学んでいきます。この例では、ピザとドーナッツの複数枚の画像からピザの特徴とドーナッツの特徴の違いをとらえていくことで正しく区別できるようにしていきます。図の例のような場合だと、▽のような先がとがっているものがピザで、丸い形で中にももう一つ丸い形が含まれているものがドーナッツというようなルールを、機械学習が画像の特徴として学んでいきます。K平均法などの教師なし学習アルゴリズムではいくつのパターンに分けてあげるかについて、最初に指定する必要があります。つまり今回の例では、画像を２種類に分類するように最初に指定します。今回のケースで4つに分類するように誤った指定をしてしますと正しく分類できないため、教師なし学習では、アルゴリズムを指定する前に、いくつに分けるのかを正しく指定するために、あらかじめ入力すべき画像の特徴を理解しておく必要があります。

| æå¸«ãªãå­¦ç¿- kå¹³åæ³ï¼ã¯ã©ã¹ã¿ãªã³ã°ï¼
10
 

教師無し学習の分類の学習イメージ

半教師あり学習とは

半教師あり学習とは、教師あり学習と教師なし学習を組み合わせたものです。

機械学習で学習用のデータを準備する場合、学習させるデータを多量に準備するだけでも手間がかかります。教師あり学習では、それぞれのデータにさらに正解のラベルまで準備する必要があり、教師あり学習のデータを準備するのには非常に労力がかかります。たとえば1万枚のピザとドーナッツの画像があったとします。1万枚の画像に対してピザかドーナッツかの正解のラベルを付けるのは大変手間がかかりますよね。そこで、データの一部であるたとえば1000枚の画像に対して教師ありデータを準備し、それ以外の9000枚のデータは教師なしデータとすることで、正解のラベルを付けるという作業は大幅に減らすことができます。半教師あり学習では、最初にぞれぞれのデータごとに個々の学習を行い、最後にこの２つを組み合わせることで、すべてのデータに対して正しく正解のラベルまで答えられるようにします。これを応用し、半教師あり学習は教師なしデータに対して正解のラベルを付けた教師ありデータにするようなタスクにも利用されています。

強化学習とは

強化学習は、今まで説明したそれぞれの学習方法と違い、予測した結果に対してそれが正しかったか誤っていたかを教えていくことで、できるだけ正しく予測するようにトライアンドエラーを繰り返していくという学習方法になります。実際には報酬と呼ばれる、うまくいけばポイントがあがり、失敗した場合には罰としてポイントが下がるような仕組みで学習を行っていきます。以前話題になった囲碁の例で説明すると、最終的に勝利した対局における手に関しては例えば＋１ポイントを与え、負けた場合は－１ポイントを与えるといった報酬を設計しておきます。すると、何万回と対局を重ねるうちにどういう局面でどういう手を打てば勝利につながるか学習していくことが可能です。

1.3 機械学習アルゴリズムのイメージ

本節では、機械学習アルゴリズムがどのように学習を進めていくのかについて説明したあと、ディープラーニングの基本モデルを使って実際のプログラミングの流れと学習結果について見ていきます。

まず機械学習が具体的にどのようなことをやっているかについて知るために、機械学習のHello Worldにあたる以下の例について説明します。

æ©æ¢°å­¦ç¿ã®"Hello Worldâ
!12
x = -2, -1, 0, 1, 2, 3, 4
y = -3, -1, 1, 3, 5, 7, 9
y = f(x)
 

機械学習のHello World

上図のxがデータでyが答えとなるデータセットがあり、このxとyの間にどのような関係が成り立つか考えてみます。いかがでしょうか。答えはy = 2x + 1です。人がxとyの関係性を見つけていく場合には例えば、xが1ずつ増えていくとyは2ずつ増えているからy = 2x の関係があると読み取り、つぎにxが0のときyが1なので、+ 1だというような推論をしていきます。このような思考によりxからyへのマッピング、つまり関数を得ることができます。

同様の問題を機械学習で解くというのはやや大袈裟ではありますが、試しに流れを見てみましょう。まず下図の青い点のデータに対して、モデルの初期値として適当に緑色の線のような直線を準備します。ここで直線にしている理由は、図を見てわかる通り青の線は基本的に直線のモデルで正しく予測できるだろうという仮定があり、y\_ = ax + b の形をした一次関数でａとbがパラメータで表現できるという前提が含まれるからです。

æ©æ¢°å­¦ç¿ã®ã¢ãã­ã¼ã
â¢é©å½ã«ã¢ãã«ãåæå (y_=ax+b)
!14
 

機械学習の初期状態

次に、モデルへのデータ入力で得られた答えと実際に推定したい青のデータ点の間の誤差を計算します。誤差の計算方法にはさまざまなものがありますが、数学的な裏付けがある二乗誤差（MSE: Mean Squared Error）がよく用いられます。下図の赤のそれぞれの矢印線で示される2点の間の長さの２乗の和の平均値を誤差として扱います。実際の青のデータ点に近づけるために、その誤差を小さくするように少しだけパラメータを更新していきます。

æ©æ¢°å­¦ç¿ã®ã¢ãã­ã¼ã
â¢ é©å½ã«ã¢ãã«ãåæå (y_=ax+b)
â¢èª¤å·®ï¼æå¤±ï¼ãè¨ç® (L=1/N Î£(y-y_)2)
!15
 

機械学習でパラメータ更新のためのアプローチ

パラメータを更新したら例えば下図のようになります。

æ©æ¢°å­¦ç¿ã®ã¢ãã­ã¼ã
â¢ é©å½ã«ã¢ãã«ãåæå (y_=ax+b)
â¢ èª¤å·®ï¼æå¤±ï¼ãè¨ç® (L=1/N Î£(y-y_)2)
â¢èª¤å·®ãâ¼©ãããªãããã«ãã©ã¡ã¼ã¿ (a, b)
ãå°ãæ´æ° (a â a - Î· âL/âa)
!16
 

機械学習のパラメータ更新

それ以降、再度誤差を計算してパラメータを更新するというプロセスを延々と繰り返すことで、y = 2x + 1にどんどん近づいていくということをやっています。このような背景から、東京大学の松尾豊先生は”ディープラーニングは最小二乗法のお化けのようなもの”との表現されていたのかと思います。

今までの説明を、Googleが開発しているTensorFlowのKeras APIというフレームワークを用いてプログラムにしたものが以下になります。

!19
 

実行可能なサンプルプログラムは本書のサンプルプログラムの参照ページからダウンロードできます。本サンプルプログラムをJupyter Notebookなどにインポートして実行してみてください。

最初のコードで以下のように使用するモジュールのインポートを実行します。

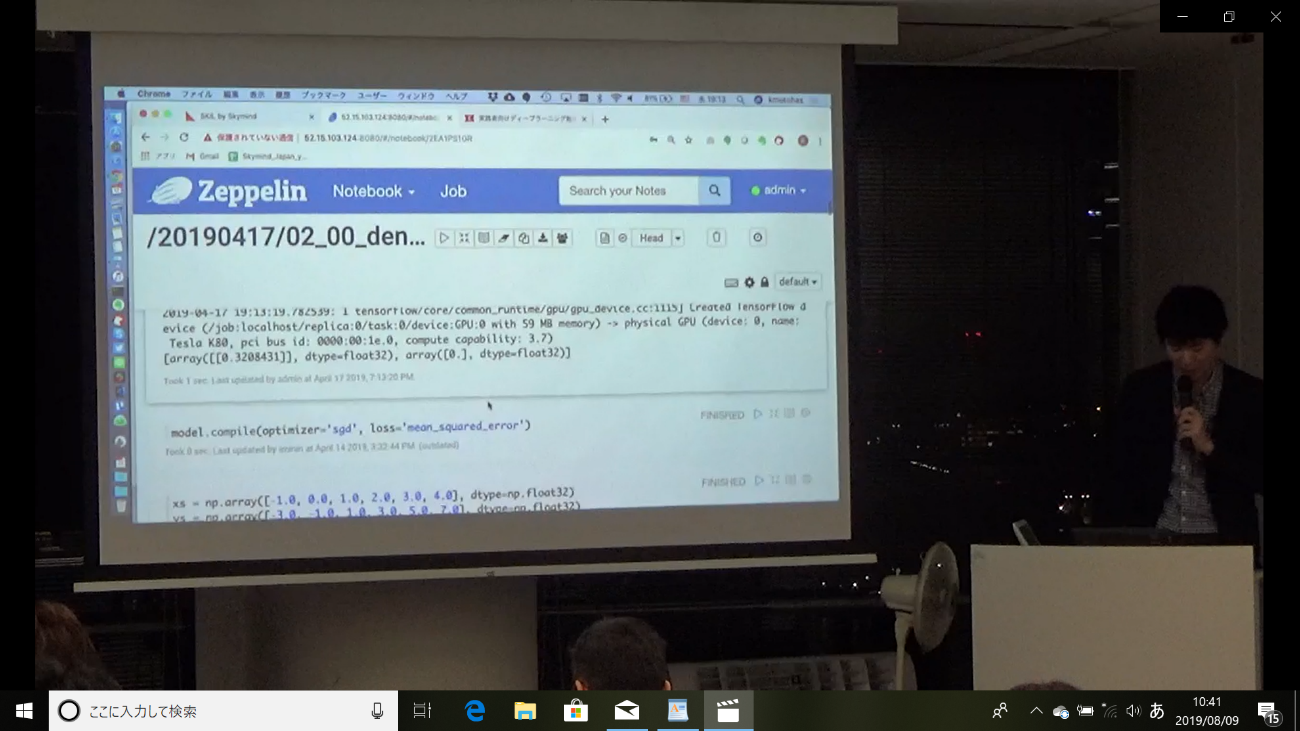
|  |
| --- |
| import tensorflow as tf  import numpy as np |

つぎに先ほどの例のax + b にあたる機械学習のモデルを準備します。

!20
x
a
+ b
y_
 

上図の例は全結合ニューラルネットワークと呼ばれる形式で、ユニットが１つで入力の型が１のネットワークを定義します。全結合ニューラルネットワークの構成要素はKerasではDense Layerと名付けられています。初期値としてはaのパラメータには0.3208431、bには0などという適当な値が入ります。

実際には実行するたびにランダムな値が入るため、読者の方がサンプルコードを実行される際には、異なるパラメータの値となりますので、ご注意ください。

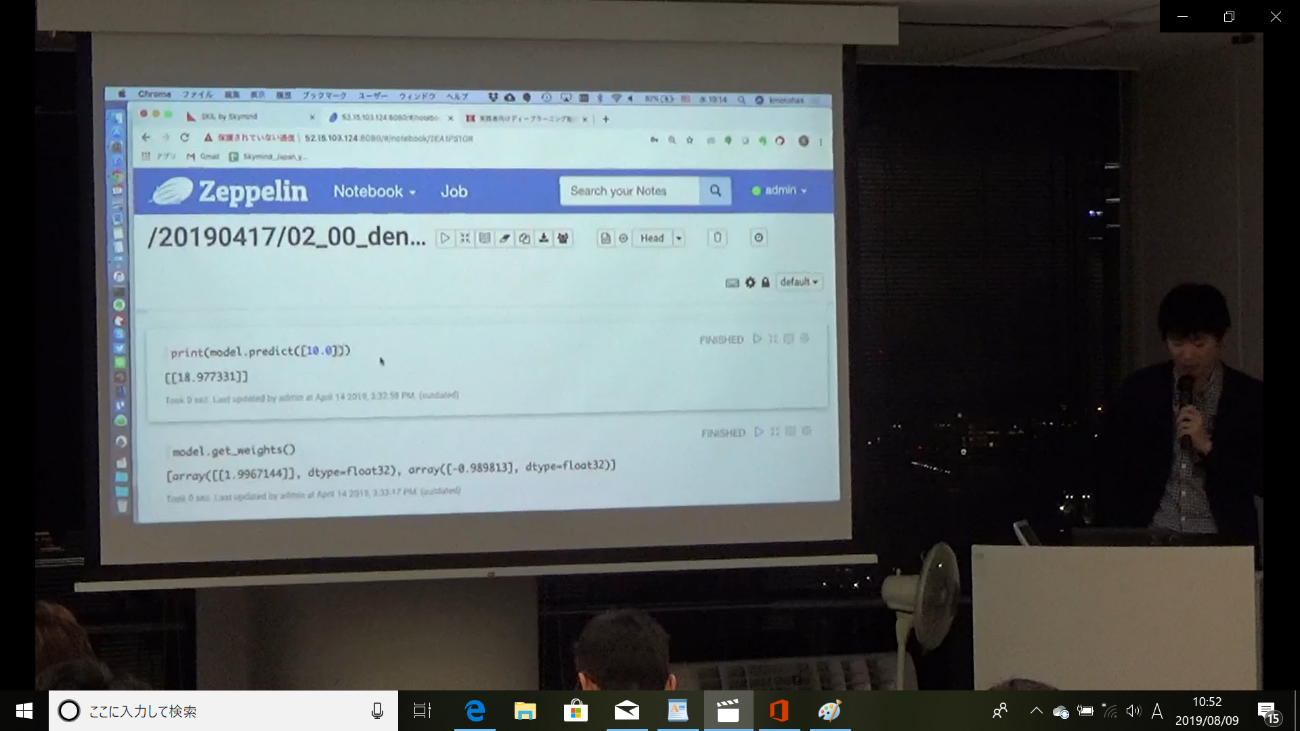


つぎにこのモデルをどういう風に最適化していくかを決め、誤差関数を何にするか定義します。最適化手法はここでは基本的な確率的勾配降下法（SGD: Stochastic Gradient Decent）を用いることにします。誤差関数は前述の通り平均二乗誤差を用います。

!21
 

そこで出来上がったモデルに対してデータを入力します。データの形式は図のように配列になっています。

つぎにfitという関数を呼ぶことで、学習が行われます。誤差（損失、lossとも）が小さくなるように同様の処理が繰り返されていきます。



上図のように最終的に学習が終わったモデルに対して、10.0という値を入力してあげると、正解値の19に近い18.98266という値が返ってきていることが理解いただけると思います。

ただしニューラルネットワークは確率的にパラメータを更新していくため、完全に19という正解値に到達することはほぼありませんが、今回の例で、学習が進んでいく様子を理解いただけたと思います。

今回のサンプルプログラムではKerasを用いていますが、Kerasにかぎらず近年の主なディープラーニングの開発フレームワークを使うことで、今回の例のように一見すると複雑ですがよく見ていただくとわかる通り、簡単にモデルを定義することができます。

1.4 全結合ニューラルネットワーク

ディープラーニングのモデルにもさまざまなものが存在しますが、まず一番基本的な全結合ニューラルネットワーク（Dense Layer）について説明します。２章以降で紹介するいくつかのアルゴリズムの理解の助けにもなる情報ですのでしっかり理解しましょう。

ニューラルネットワークの解説では先ほどの図の右上のようなニューロン同士が密に結合している図がよく出てくると思いますが、この図のように表現されるニューラルネットワークが全結合ニューラルネットワークと呼ばれるものです。内部で何を行っているかというと、単純に「重み」×「x」＋「バイアス」のような ax + bを多次元的に計算しているだけです。数学的に言うと、線形的に結合して和をとっているだけです。

さきほど最適化のためのアルゴリズムを設定すると説明しましたが、ソースコードにあったsgdがどのようなものかを説明します。この最適化手法は確率的勾配降下法（SGD）と呼ばれるものです。ディープラーニングの世界で最適化のアルゴリズムとしてデファクトスタンダードとなっています。

最適化を行っていくには重みなどのパラメータを更新していきますが、具体的な重みの更新を数式で表したものが図のようにとなります。それは誤差関数（損失関数、loss functionとも)のパラメータでの微分をとってさらに学習率と呼ばれる適当な係数をかけて、その値を新しい重みの値にすることを表しています。

| æé©åã¢ã«ã´ãªãºã é¸æ
ç¢ºççå¾ééä¸æ³ SGD, stochastic gradient descent.
ããã¡ã¯ãã¹ã¿ã³ãã¼ã
ï¼æ¹è¯çï¼momentum, Nesterovã®å éæ³, RMSProp, Adam, ...ï¼
43
 

最適化アルゴリズムの選択

この数式が具体的にどのようなことをやっているかについて上の図を用いて説明します。図の青の曲線が、重みパラメータwに対する誤差関数だと思ってください。たとえばパラメータが図の一番左端の点にあるとき、その曲線の傾きを計算すると負の値をとります。誤差を最小にするには、パラメータが図の曲線の一番へこんだ場所にいって欲しいわけです。つまり、左端の赤い点から右側に移動してくれればよいのです。この図では、左端の赤い点における傾きは右下がりなので負の値になっているため、重みを右に動かしたい場合は負の傾きの値を引き算してあげると重みの値が大きくなる、つまり赤い点が右側へ移動していきます。

実際に最適化を実行していく場合、確率的勾配降下法はあまりにもシンプルな手法なので、その改良版として、モーメンタム、Nesterovの加速法、RSMProp、Adamなどさまざまなものが考案されています。2019年の3月にも国際会議ICLR2019で採択された最適化の最新のアルゴリズムとしてAdaBoundが出てくるなど、日々研究が進んで進化している分野ですので、新たなものについても情報を収集するようにぜひ心がけてください。ただし基本的にはある程度高速にパラメータが収束することが経験的に知られているAdamを使えばよいと思います。たいていのディープラーニングフレームワークではAdamは簡単に用いることができます。先程のコードの’sgd’の部分を’adam’と書き換えればOKです。

・勾配降下法のイメージ

å¾ééä¸æ³
!23
ãâ½¬é ãã§â¾åã®å¾éæå ±ã®ã¿ãä½¿ã£ã¦â¼­ã®é ä¸ãâ½¬æããããªãã®ã
å­¦ç¿ç Î· ã¯æ­©å¹ã®ã¤ã¡ã¼ã¸ï¼Î·â¼©ï¼ããâ¾ãÎ·â¼¤ï¼å·¨â¼ã®â¼æ­©ï¼
https://twitter.com/momiji_fullmoon/status/1110...

勾配降下法のイメージ

今までの説明内での目的は誤差の最小化でしたが正負の符号を変えるだけで最大化問題になるので、次の説明では最大化問題を解きたいと思って読み進めてください。

勾配降下法とは「目隠しで足元の勾配の感覚のみを頼りに山の頂上を目指すようなもの」と喩えることができます。頂上を目指す場合には、目を閉じてそれぞれ場所の足元の感覚から上り坂だと思われる方に進むということを延々と繰り返すようなものだということです。図の数式に出てくるギリシャ文字のη（イータ）という記号は、学習率と呼ばれるパラメータですが、これは歩幅のイメージです。たとえばこのηの値が0.001のように非常に小さい値の場合には、すり足のようにゆっくり慎重に進んで、大きな値であれば大股で進んでいくという感じです。誤差関数は大抵きれいな二次曲線の形をしているわけではありません。実際の山のように、ある一帯で見た場合には最小値となる、極小値と呼ばれる局所的に一番低い場所がたくさんあるゴツゴツとした形になっているでしょう。そのため目標となる最小値の場所ではない極小値のところで落ち着かないように、学習し始めのうちはそういう場所は気にせず飛び越えてもらいたいのです。それを実現するために、最初は学習率を大きめにして大股で歩いて行って、頂上に近づいてきたら学習率を小さい値にして歩幅を小さくするようにスケジューリングして実行していくということがよく行われています。

・使用するデータセットの紹介

次の章で説明するCNNの話に近づいていくために、Fashion MNISTデータセットというものをここで紹介します。ディープラーニングの基礎的な話を聞く場合、手書き文字の認識の話がよく話題になると思います。Fashion MNISTのデータセットは手書き文字より少し複雑になったもので、7万枚の画像データが含まれています。以下の図のようにTシャツ、コート、バッグ、スニーカー、などファッショングッズに関する10カテゴリのピクトグラムのような小さな画像が準備されています。それぞれの画像の大きさは28×28ピクセルで、よく実験やベンチマークで用いられるデータセットです。

Fashion MNIST Dataset
â¢ 7ä¸ç»å
â¢ 10ã«ãã´ãª
â¢ 28Ã28 pixels
â¢ å®é¨â½¤ãã¼ã¿ã»ãã
!24
https://github.com/zalandoresearch/fashion-mnist
 

Fashion MNISTデータセット

このデータセットを使って、これから全結合ニューラルネットワークによる各カテゴリに分類するモデルを作っていきます。

・全結合ニューラルネットワークの実行例

下のサンプルコードを参照しながら全結合ニューラルネットワークの具体的な実装について説明します。

以下ではサンプルコードの赤枠で示されている部分について説明をします。

サンプルコードの最初の部分では、線形代数などの数値計算ライブラリであるnumpyや機械学習用のtensorflowや可視化用のmatplotlibと呼ばれるモジュールをインポートします。

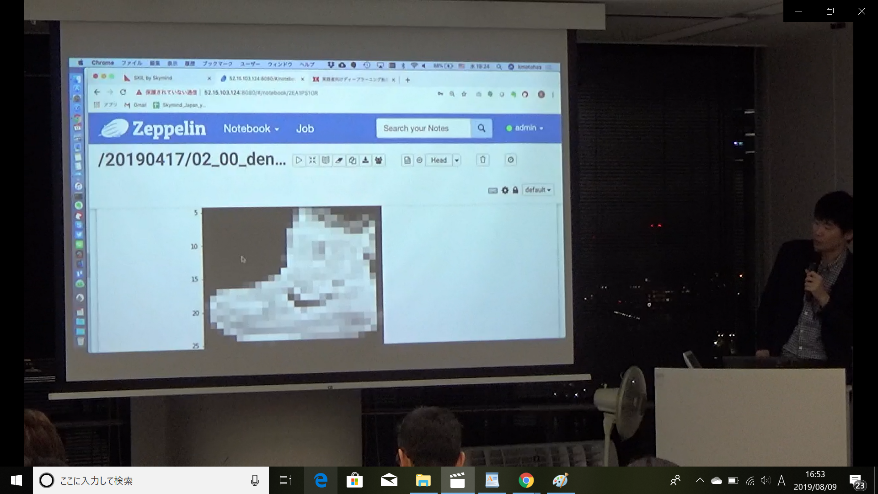
次にデータセットをロードします。Fashion MNISTデータセットは非常に有名なものでよく使われるため、TensorFlowにも含まれています。ここではTensorFlowのライブラリを使ってデータセットを読み込みます。そうすることで、訓練用の画像と訓練用のラベルのペアが多次元配列に含まれたものが事前に準備され、さらに性能評価用にテスト用の画像とテスト用のラベルのペアも用意されます。こちらのデータセットの画像はカラーではなくグレースケールなので、画像の行列の中身は各ピクセルに対して0（黒）～255（白）の範囲の輝度と呼ばれる明るさを数値化したものが入っています。このままの値を用いてしまうと、数値が大きすぎて訓練がうまくいかなくなる場合があるため、数値を0～1の間の値に変換します。このようにデータを利用しやすい形に整理したり変換したりすることは正規化と呼ばれます。

!27
 

以下で具体的な画像の中身がどのようになっているか確認します。

train\_imagesという訓練用の画像は、6万個の画像があって、高さ28ピクセル、幅28ピクセルの形の配列になっています。ラベルの方は同じく6万個あって0～9の10個の値のどれかが入っていて、たとえば0番目の画像では9つまりアンクルブーツというラベルがついていることを意味しています。

実際に最初の画像を可視化してみると以下のように非常に粗いモザイクがかかったようなブーツであることがわかります。



Fashion MNISTデータセットのブーツの画像例

上記のデータセットに対してモデルを組み立てていきましょう。

まず28×28ピクセルの形の画像を全結合ニューラルネットワークに入力する必要があるため、以下のようにFlattenと呼ばれる操作で1列の784次元のベクトルに変換します。具体的には二次元行列の0番目の行の要素を縦に並べ、続けて1番目の行のデータをまた縦に並べるという作業を28行分繰り返して、全部で1列の784次元のデータにします。今回のモデルも全結合ニューラルネットワークで構成します。図のように、128個のニューロン（ユニット）を中間の層に配置して、その後ろに分類を行うための10個のニューロンを配置するというモデルを作っています。

!28
â¦
â¦â¦â¦â¦
â¦
â¦
Flatten: (28, 28) => (784)
0 1 2 3 4 5 6 7 125 126 127
0 1 2 3 4 5 6 7 8 9
 

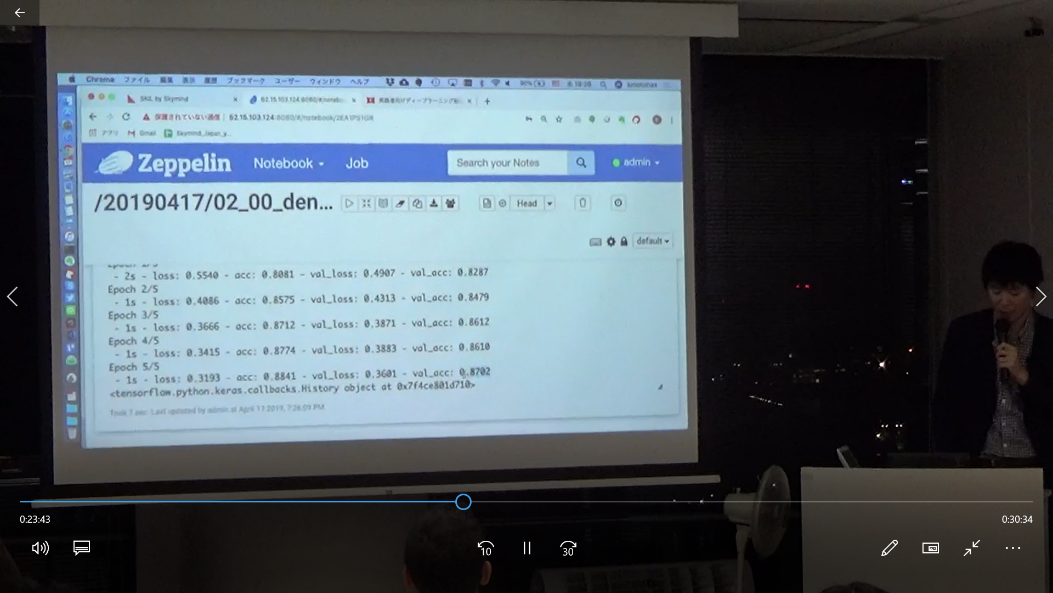
画像データのニューラルネットワークへの入力イメージ

こちらのコードのようにKerasはわかりやすくそれぞれのモジュールを配列として3つ並べるだけで図のようなニューラルネットワークが完成します。

つぎにこちらのモデルに対して最適化のアルゴリズムとしてはAdamを用いることにします。損失関数としては詳しい説明は割愛しますが、分類の問題の損失関数を評価する際に一般的に用いられる交差エントロピー(cross entropy)というものを用います。下図の赤枠のコードの部分が具体的なプログラムコードです。

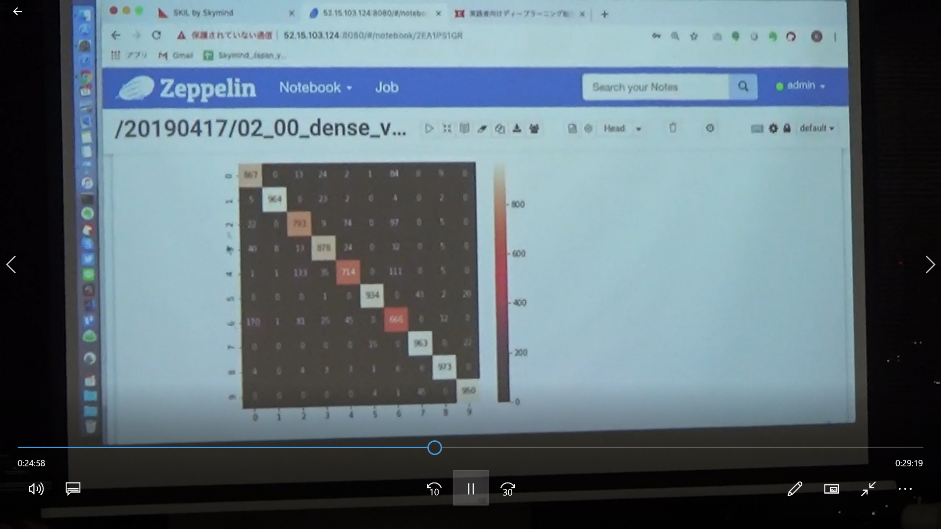
!27
 

今までのプログラムで設定したデータセットとモデルを使って実際に学習を実行させるために、前に説明したfitというメソッドを用いてtrain\_imagesとtrain\_labelsというデータを入力として、訓練を行います。実際にこちらを実行すると以下の出力結果のように、だんだんと正解率(accuracy)が上がっていっているのが確認できます。このようにある程度ループを回すことで、たとえばドレスの画像を入れたら87パーセントの確率でこれはドレスですといったことを推論してくれるようなモデルができあがります。



つぎに評価がどのようになされるかをイメージしていただくために、混合行列（confusion matrix）について説明します。混合行列とはどれをどれくらい正解してどれくらい間違っているかを確かめるための表です。

以下の図が混合行列をヒートマップ化したものです。図の横軸が正解のラベル、縦はモデルが予測したラベルを示しています。正解ラベルが0で予測も0であれば正しく予測できている、というような行列になっていて、図全体でみてみると、対角線上にある値が正解した数で、それ以外の値が間違った数になっています。対角線以外は、それぞれどの正解ラベルがどのラベルと誤って予測されたのかが示されています。どこらへんに問題があるのかとかどこらへんが間違いやすいのかについて、混合行列のそれぞれのラベルを見ながら探っていくことができます。



混合行列の可視化例

・全結合ニューラルネットワークを画像に用いる場合の欠点

全結合ニューラルネットワークを画像に用いる場合の欠点について触れておきます。全結合ニューラルネットワークは、入力ベクトルの全要素のつながりを見ています。画像の場合には基本的に、例えば左上側に位置するピクセルに対して右下側のピクセルにはあまり関係性がなく画像の性質で考えると無視してほしいものになりますが、全結合ニューラルネットワークではそのような無関係なものまで関係性を見つけようと試みます。学習していく中で、あるニューロンと別のニューロンには関係性がないのでつながりの重みがゼロであるといようにもともと明らかであった無意味なパラメータまで頑張って更新していくことになります。このようなやり方は計算コストがかかるのと、そもそもあまりにも相関が複雑になりすぎて学習が進まないということが起こりえます。全結合ニューラルネットワークは住宅の価格予測のようにすでに作成済みの特徴量の組み合わせでこの家の価格はいくらくらいになるのかを予測するときには、明確にそれぞれの特徴量に相関を比較的簡単に見つけられるので有効ですが、画像ではそうはいきません。画像ではむしろ各ピクセル間の大域的な関係性よりも、角が丸っこいだとかシャープだとか局所的な情報の方が特徴を掴むのに重要です。ピクセルの近い部分の関係性を学習し画像の特徴量を抽出するのに適したアルゴリズムとして代表的なものに畳み込みニューラルネットワーク（Convolutional Neural Network: CNN）というものがあります。

では早速次章で畳み込みネットワークについて見ていきましょう。