CS231n Convolutional Neural Networks for Visual Recognition

5.ニューラルネットワーク(Part.1)

序論

脳のはたらきに例えなくても，ニューラルネットワークを導入することは可能です．線形分類の項では，画像に含まれるさまざまな視覚的カテゴリーの判別スコアを，
$s = Wx$(ただし，Wは行列で，xは，画像のすべてのピクセルデータを含む入力列ベクトル)
CIFAR-10の場合，xは[3072x1]の列ベクトルであり，Wは [10x3072] の行列であり，出力スコアは10 個の分類スコアのベクトルとなります．
ニューラルネットワークの例では，線形分類の時とは変わり，$s = W_2\max(0,W_1x)$を計算します．ここで，W_1は，例えば，画像を100次元の中間ベクトルに変換する[100x3072]行列などが考えられます．関数max(0,–)は，要素ごとに適用される非線形性です．
非線形性にはいくつかの選択肢がありますが(以下で検討します)，この選択肢は一般的なもので，単純に0以下のアクティベーションをすべて0にしきい値化します．最後に，行列W_2は[10x100]の大きさになり，クラススコアと解釈される10個の数字が再び得られます．この非線形性は，計算上非常に重要であることに注意してください．もし，この非線形性を残しておけば，2つの行列は1つの行列に折りたたまれ，予測されるクラススコアは再び入力の線形関数となります．非線形性は，私たちが微動だにしないところです．パラメータW_2, W_1は確率的勾配降下法で学習され，その勾配は連鎖律で導かれます(バックプロパゲーションで計算されます)．
3層構造のニューラルネットワークは，数式で表現するならば$s = W_3\max(0,W_2\max(0,W_1x))$のようになり，$W_3, W_2, W_1$のすべてが学習されるパラメータとなります．中間の隠れベクトルの大きさは，ネットワークのハイパーパラメータで，その設定方法は後ほど説明します．では，これらの計算をニューロンやネットワークの観点からどのように解釈するかを考えてみましょう．

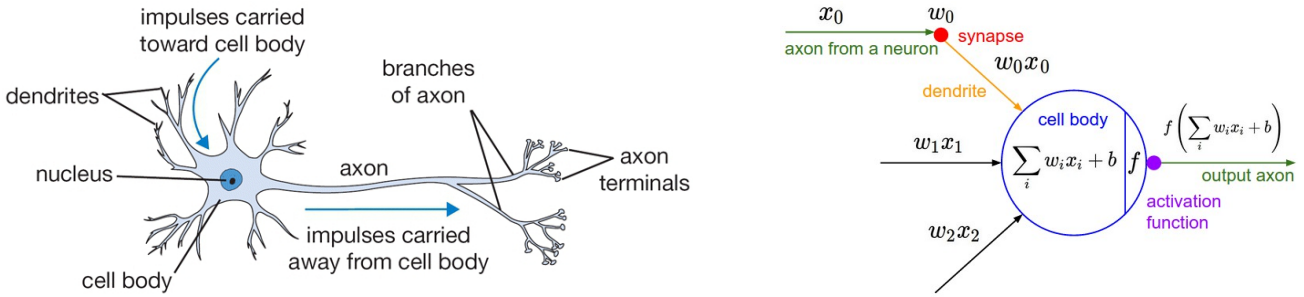
ニューロンのモデリング

ニューラルネットワークの分野は，元々は生物学的な神経システムをモデル化するという目的に触発されていましたが，その後，エンジニアリングの問題となり，機械学習のタスクで良い結果を出すことができるようになりました．それでも，この分野の大部分が触発されてきた生物学的システムについて，非常にシンプルながらもハイレベルな説明で議論を始めます．

生物学的刺激とつながり

脳が構成されている基本的な単位はニューロンです．人間の神経系には約860億個のニューロンが存在し，それらは約10^{14}~10^{15}個のシナプスで接続されています．下の図は，生物学的なニューロンの概略図(左)と，一般的な数学的モデル(右)を示しています．各ニューロンは，樹状突起から入力信号を受け取り，その(単一の)軸索に沿って出力信号を生成します．軸索は最終的に分岐し，シナプスを介して他のニューロンの樹状突起に接続される．ニューロンの計算モデルでは，軸索を伝わる信号(例えばx_0)は，そのシナプスでのシナプス強度(例えばw_0)に基づいて，相手のニューロンの樹状突起と乗算的に相互作用をします．(例えばw_0x_0).

考え方としては，シナプスの強度(重みw)は学習可能であり，あるニューロンが他のニューロンに与える影響の強さ(およびその方向：興奮性(正の重み)または抑制性(負の重み))を制御するという考え方です．基本的なモデルでは，樹状突起が信号を細胞体に伝え，そこですべての信号が合計されます．最終的な和がある閾値を超えると，そのニューロンは発火し，軸索に沿ってスパイクを送ることができます．計算モデルでは，スパイクの正確なタイミングは重要ではなく，発火の頻度のみが情報を伝達すると仮定しています．このレートコードの解釈に基づいて，ニューロンの発火率を，軸索に沿ったスパイクの頻度を表す活性化関数fでモデル化します．これまで，活性化関数の一般的に用いられている関数はシグモイド関数σです．これは，実数値の入力(和の後の信号強度)を受け取り，0と1の間の範囲になるようにそれを正規化するからです．これらの活性化関数の詳細については，このセクションの後半で見ていきます．



生物学的なニューロン(左)とその数学的モデル(右)を図式した

```
class Neuron(object):
  # ...
  def forward(self, inputs):
    """ assume inputs and weights are 1-D numpy arrays and bias is a number
"""
    cell_body_sum = np.sum(inputs * self.weights) + self.bias
    firing_rate = 1.0 / (1.0 + math.exp(-cell_body_sum)) # sigmoid
activation function
    return firing_rate
```

単一のニューロンを順方向に伝搬させるコードの例は次のようになります．
言い換えるならば各ニューロンは入力とその重みのドット積を行い，バイアスを加え，非線形性(または活性化関数)，ここではシグモイド$\sigma(x) = 1/(1 + e^{-x})$を適用します．活性化関数の違いについては，このセクションの最後で詳しく説明します．

粗のモデル．ここで強調しておきたいのは，この生物学的ニューロンのモデルは非常に「粗」であるということです．例えば，ニューロンにはさまざまな種類があり，それぞれが異なる特性を持っています．生物学的ニューロンの樹状突起は，複雑な非線形計算を行います．シナプスは単なる1つの重みではなく，複雑な(非線形な)活性化関数で構成されています．多くのシステムでは，出力スパイクの正確なタイミングが重要であることが知られており，レートコード近似が成り立たないことが示唆されています．このように，ニューラルネットワークはさまざま面で簡略化されており本物の脳と類似しているなどと言おうものなら，脳神経科学のバックグラウンドを持つ人たちか

ら呪いの言葉を浴びせられることを覚悟しなければなりません．興味のある方は，このレビュー(pdf)や，より最近のこのレビューをご覧ください．

線形分類器としてのニューロン単体

モデル・ニューロンの前進計算の数学的形式は，皆さんにもなじみがあるかもしれません．線形分類器で見たように，ニューロンは，入力空間の特定の線形領域を「好き」(活性化が1に近い)または「嫌い」(活性化が0に近い)にする能力を持っています．したがって，ニューロンの出力に適切な損失関数を設定すれば，1つのニューロンを線形分類器に変えることができます．

2値のソフトマックス分類器．例えば，$\sigma(\Sigma_i\ w_i x_i + b)$は一方のクラスの確率$P(y_i = 1 \mid x_i; w)$と解釈できます．他のクラスの確率は$P(y_i = 0 \mid x_i; w) = 1 - P(y_i = 1 \mid x_i; w)$となります． となり，これらの合計は1になるはずです．このように解釈すると，交差エントロピー損失を線形分類の項で見たように定式化することができ，これを最適化すると，2値のソフトマックス分類器(ロジスティック回帰とも呼ばれる)が得られます．シグモイド関数は0~1の間に制限されているので，この分類器の予測は，ニューロンの出力が0.5より大きいかどうかに基づいています．

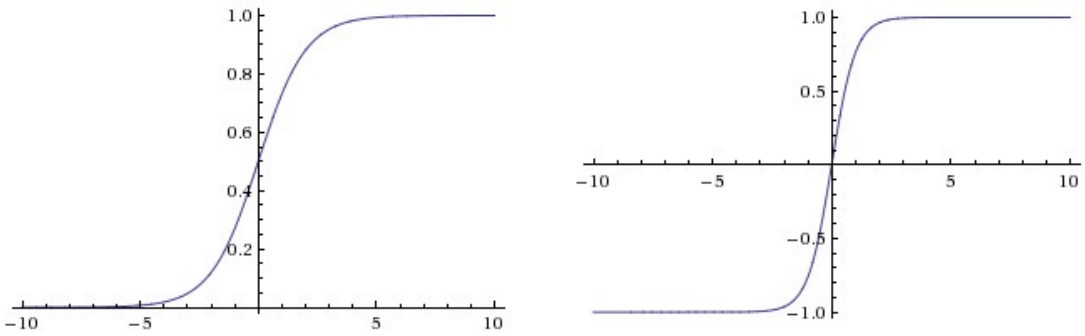
2値のSVM分類器．あるいは，ニューロンの出力にmax-marginヒンジ損失を付けて，2値のSVMになるように訓練することもできます．

正則化解釈．SVMまたはソフトマックスの両方のケースにおける正則化損失は，生物学的には，パラメータ更新のたびにすべてのシナプス重みwをゼロに向かって駆動する効果があるので，緩やかな忘却と解釈することができます．

☞単一のニューロンは，2値の分類器を実装するために使用することができます(例：2値のソフトマックスやバイナリSVM分類器)

よく使われる活性化関数

すべての活性化関数(または非線形関数)は，1つの数値を受け取り，その数値に対して一定の数学的操作を行います．実際に使用する可能性のある活性化関数はいくつかあります．



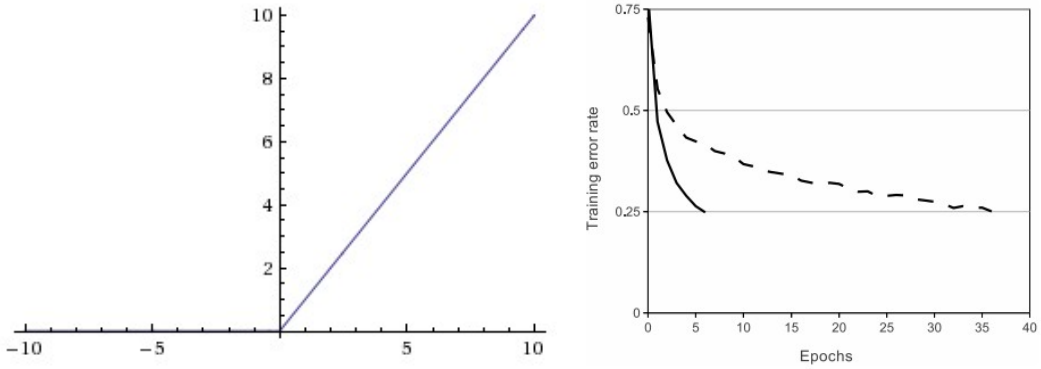
左：シグモイド活性化関数は実数を[0,1]の範囲に圧縮する．右：tanh活性化関数は実数を[-1,1]の範囲に圧縮する．tanh非線形性は，実数を[-1,1]の範囲に圧縮する

シグモイド関数． シグモイド非線形関数は，上図左の$\sigma(x) = 1/(1 + e^{-x})$になります．前節で述べたように，シグモイド関数は，実数を0から1の範囲に「圧縮」するもので，大きな負の数は0に，大きな正の数は1になります．シグモイド関数は，全く発火しない状態(0)から，想定される最大周波数で完全に飽和した発火状態(1)まで，ニューロンの発火率として解釈できるため，歴史的に頻繁に使用されてきました．実際には，シグモイド非線形性は最近では人気がなくなり，ほとんど使用されていません．2つの大きな欠点があるからです．

・シグモイドは勾配を飽和させてしまう．シグモイド・ニューロンの非常に望ましくない特性は，ニューロンの活性化が0または1のどちらかのテールで飽和すると，これらの領域での勾配がほとんどゼロになることです．バックプロパゲーションの際には，この(局所的な)勾配が，目的全体のこのゲートの出力の勾配に掛けられることを思い出してください．したがって，局所的な勾配が非常に小さい場合，実質的に勾配を「殺す」ことになり，ニューロンを介してその重みや再帰的にそのデータにほとんど信号が流れなくなります．さらに，シグモイドニューロンの重みを初期化する際には，飽和を防ぐために細心の注意を払う必要があります．例えば，初期の重みが大きすぎると，ほとんどのニューロンが飽和してしまい，ネットワークはほとんど学習しなくなってしまいます．

・シグモイドの出力はゼロ中心ではない．これは，ニューラルネットワークの後続の処理層のニューロンがゼロ中心ではないデータを受け取ることになるので，望ましくありません．このことは，勾配降下法のダイナミクスに影響を与えます．なぜなら，ニューロンに入力されるデータが常に正である場合．(例えば，$f = w^T x + b$の要素ごとに$x > 0$)．バックプロパゲーションの際の重みwの勾配は，すべて正になるか，すべて負になるかのいずれかになるからです(式全体の勾配に依存するf)．これにより，重みの勾配更新に望ましくないジグザグの動きが生じる可能性があります．しかし，これらの勾配がデータのバッチ全体に渡って加算されると，重みの最終的な更新は可変の符号を持つことになり，この問題が多少緩和されることに注意してください．したがって，この問題は不便ではありますが，上記の飽和活性化の問題に比べれば，それほど深刻な影響はありません．

Tanh．上の画像の右にあるのがtanhの活性化関数です．実数を[-1, 1]の範囲に圧縮します．シグモイド・ニューロンと同様に，その活性化は飽和しますが，シグモイド・ニューロンとは異なり，その出力はゼロ中心です．したがって，実際にはシグモイド活性化関数よりもtanh活性化関数の方が常に好ましい．また，tanhニューロンは単純にスケールされたシグモイドニューロンであり，特に次のことが成り立ちます．$\tanh(x) = 2\sigma(2x) - 1$



左：Rectified Linear Unit (ReLU)活性化関数，x < 0のときは0，x > 0のときは1の傾きを持つ線形となる．Krizhevskyらの論文(pdf)からのプロットで，ReLUユニットではtanhユニットに比べて収束性が6倍向上していることがわかる．

ReLU．Rectified Linear Unitは，ここ数年で非常に人気のあるユニットです．これは，関数$f(x) = \max(0, x)$を計算するものです．言い換えれば，活性化は単純にゼロで閾値化されます(上の左の画像を参照)．ReLUの使用にはいくつかの長所と短所があります．

・(+) sigmoid/tanh関数と比較して，確率的勾配降下法の収束を大幅に加速することがわかった(例えば，Krizhevskyらの論文では6倍)．これは，線形で飽和していない形式によるものだと主張されています．

・(+) 高負荷な演算(指数など)を伴うtanh/sigmoidニューロンに比べ，ReLUは活性化の行列をゼロで閾値処理するだけで実装できる．

・(-) 残念ながら，ReLUユニットは学習中に壊れやすく，「死んで」しまうことがあります．例えば，ReLUニューロンに大きな勾配がかかると，重みが更新され，ニューロンがどのデータポイントでも二度と活性化しなくなることがあります．そうなると，そのユニットを流れるグラジエントは，その時点から永遠にゼロになります．つまり，ReLUユニットはデータマニホールドから外れてしまうため，トレーニング中に不可逆的に死んでしまう可能性があるのです．例えば，学習率を高く設定しすぎると，ネットワークの40%が「死んだ」状態(トレーニングデータセット全体でアクティブにならないニューロン)になってしまうことがあります．学習率を適切に設定することで，この問題は少なくなります．
Leaky ReLU．これは，「瀕死のReLU」問題を解決する一つの試みです．x < 0の時に関数がゼロになる代わりに，Leaky ReLUは小さな負の傾き(0.01程度)を持ちます．つまりこの関数は，$f(x) = \mathbb{1}(x < 0)(\alpha x) + \mathbb{1}(x >= 0)(x)$と計算されます．ここで$\alpha$は小さな定数です．この形式の活性化関数で成功したという報告もありますが，結果は常に一定ではありません．また，Kaiming He et al., 2015によるDelving Deep into Rectifiersで紹介されているPReLUニューロンに見られるように，負の領域の傾きを各ニューロンのパラメータにすることもできますが，タスク間でのメリットの一貫性は現在のところ不明です．

Maxout．重みとデータの間のドット積に非線形性が適用される関数形式$f(w^T x + b)$を持たない他のタイプのユニットも提案されています．比較的よく知られているのは，ReLUとそのリーキー・バージョンを一般化したMaxout・ニューロン(Goodfellowらが最近発表)です．Maxout・ニューロンは，関数 $\max(w_1^T x + b_1, w_2^T x + b_2)$を計算します．ReLUもLeaky ReLUも，この形式の特殊なケースであることに注意してください(例えば，ReLUの場合は$w_1, b_1 = 0$). したがって，Maxoutニューロンは，ReLUユニットのすべての利点(線形動作領域，飽和なし)を享受し，その欠点(ReLUが瀕死状態になる)はありません．しかし，ReLUニューロンとは異なり，1つのニューロンごとにパラメータの数が2倍になるため，パラメータの総数が多くなります．