
Python で 超実習ディープラーニング

実践！ 強化学習・画像認識・自然言語処理・ロボティクス



- セミナー情報
 - 開催日時 2019年03月15日（金） 9:30～16:45
 - 講師 浅川 伸一(東京女子大学情報処理センター)
-

配布資料

ニューラルネットの基礎

初じめの一歩

- Neural Networks
 - All about NN
-

おそらく排他的論理和を解く最短の python コードを以下に示しました。ソース

```
import numpy as np
X = np.array([ [0,0,1],[0,1,1],[1,0,1],[1,1,1] ])
y = np.array([[0,1,1,0]]).T
w0 = 2 * np.random.random((3,4)) - 1
w1 = 2 * np.random.random((4,1)) - 1
for i in range(500):
    l1 = np.tanh(np.dot(X,w0))
    l2 = 1/(1+np.exp(-(np.dot(l1,w1))))
    dl2 = (y - l2) * (l2 * (1 - l2))
    dl1 = dl2.dot(w1.T) * (1 - l1 ** 2)
    w1 += np.dot(l1.T, dl2)
    w0 += np.dot(X.T, dl1)
    print(l2.T) if i % 100 == 0 else None
import numpy as np
```

- 実習 2019si_kmnist_exercise002.ipynb
-

用語集

非線形性

- GRU
- LSTM
- Relu
- Sigmoid
- Tanh

結合パターン

- Fully connected
- Convolutional
- Dilated
- Recurrent
- Recursive
- Skip / Residual
- Random

最適化

- SGD
- Momentum
- RMSProp
- Adagrad
- Adam
- Second Order (KFac)

損失関数

- 交差エントロピー
- 敵対
- 変分
- 最大対数尤度
- Sparse
- L2 Reg
- REINFORCE

ハイパー parameters

- Learning Rate
- Decay
- Layer Size
- Batch Size
- Dropout Rate
- Weight init
- Data augmentation
- Gradient Clipping
- Beta
- Momentum

-
- 尤度, 交差エントロピー, 情報理論
 - 各種最適化手法, 確率的勾配降下法, AdaGrad, AdaDelta, RMSProp, Adam, 自然勾配法, ニュートン法, L-BGFS
 - 主成分分析, 特異値分解, 自己組織化, (非負)行列因子化法
 - 固有顔, フィッシャー顔, 部分空間法
 - マッカロックとピットの形式ニューロンと生理学的対応物
 - パーセプトロン学習則, 誤差逆伝播学習則, 正則化項 (L2, L1, L0)
 - 各種活性化関数, シグモイド関数, ハイバータンジェント, 整流線形ユニット, ソフトマックス関数, 劣微分
-

付録

- Gradient descent

置み込みニューラルネットワーク CNN

- 置み込みニューラルネットワークによる日本語古典籍くずし字 kminst の認識実習
- 姿勢推定デモ
- Style-based GAN

CNN の特徴として、次の 7 つを上げることができます。

1. 非線形活性化関数 (nonlinear activation functions)
2. 置込み演算 (convolutional operation)
3. プーリング処理 (pooling)
4. データ拡張 (data augmentation)
5. バッチ正規化 (batch normalization)
6. ショートカット (shortcut)
7. GPU の使用

上記 7 つの特徴を説明するのは専門的になりすぎるので省略します。一つだけ説明するとすれば最後の GPU とは高解像度でしかも処理速度を必要とするパソコンゲームで用いられるグラフィックボードのことです。詳細な画像を高速に画面に表示する必要から開発されたグラフィックボードですが、大規模なニューラルネットワークの計算でも用いられる数学が同じです。そのため、ゲーム用に開発されたグラフィックボードがニューラルネットワークにも用いられるようになりました。

置み込みニューラルネット(CNN)とは何か

本節では深層学習、特に CNN と呼ばれるニューラルネットワークについて解説します。

最初に画像処理の概略を述べる CNN が、それまで主流であった従来の手法の性能を凌駕したことはすでに述べました。CNN の特徴の一つに エンドツーエンド と呼ばれる考え方があります。エンドツーエンドとは、従来手法によるパターン認識システムでは、専門家による手の込んだ詳細な作り込みを必要としていたことと異なり、面倒な作り込みをせずとも性能が向上したことを持ちます。

エンドツーエンドなニューラルネットワークにより、次のことが実現しました。

- ニューラルネットワークの層ごとに、特徴抽出が行われ、抽出された特徴がより高次の層へと伝達される
- ニューラルネットワークの各層では、比較的単純な特徴から次第に複雑な特徴へと段階的に変化する
- 高次層にみられる特徴は低次層の特徴より大域的、普遍的である
- 高次層のニューロンは、低次層で抽出された特徴を共有している

このことを簡単に説明してみます。

我々人間は、外界を認識するために必要な計算を、生物種としての発生の過程と、個人の発達を通しての経験に基づく認識システムを保持していると見ることができます。従って我々の視覚認識には化石時代に始まる光の受容器としての眼の進化の歴史と発達を通じた個人の視覚経験が反映された結果もあります。人工知能の目標は、この複雑な特徴検出過程をどうやったらコンピュータが獲得できるかということでもあります。外界を認識するために今まで考案されてきたモデル（例えば、ニューラルネットワークサポートベクターマシンなど）は複雑です。ですがモデルを訓練するための学習方法はそれほど難しくありません。この意味で画像認識課題が正し

く動作するためのポイントは、認識システムが問題を解く事が可能なほど複雑であるかどうかではなく、十分に複雑が視覚環境、すなわち画像認識の場合、外部の艦橋を反映するために十分な量の像データを容易すことができるか否かにあります。今日のCNNによる画像認識性能の向上は、簡単な計算方法を用いて複雑な外部環境に適応できる認識システムを構築する方法が確立したからであると言うことが可能です。

下図に画像処理の例を挙げました。

我々の見ている画像



カメラに入力されたデータ

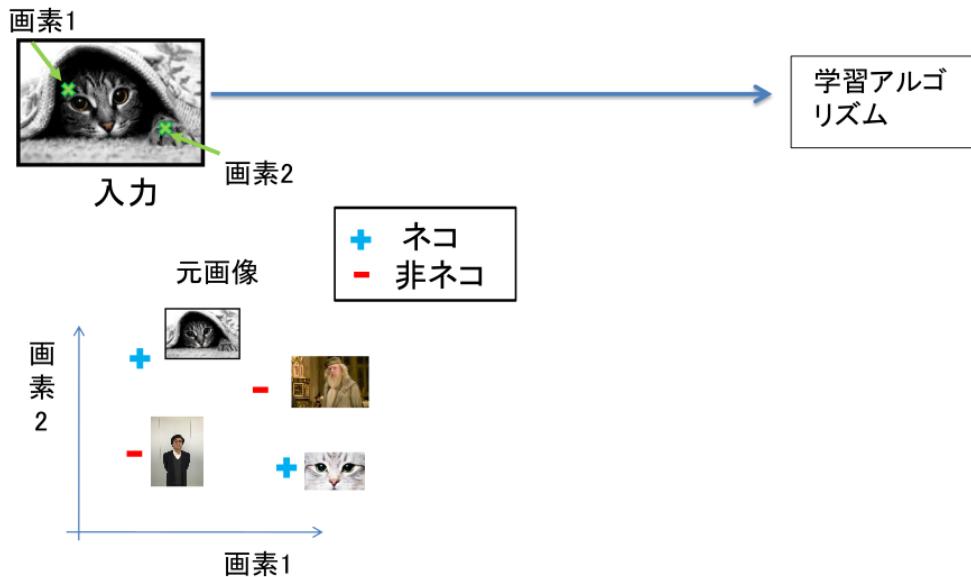
194	210	201	212	199	213	215	195	178	158	182	209
180	189	190	221	209	205	191	167	147	115	129	163
114	126	140	188	176	165	152	140	170	106	78	88
87	103	115	154	143	142	149	153	173	101	57	57
102	112	106	131	122	138	152	147	128	84	58	66
94	95	79	104	105	124	129	113	107	87	69	67
68	71	69	98	89	92	98	95	89	88	76	67
41	56	68	99	63	45	60	82	58	76	75	65
20	43	69	75	56	41	51	73	55	70	63	44
50	50	57	69	75	75	73	74	53	68	59	37
72	59	53	66	84	92	84	74	57	72	63	42
67	61	58	65	75	78	76	73	59	75	69	50

では入力画像がネコであるか否かを判断する画像認識であるとしました。我々はネコの画像を瞬時に判断できます。ですが画像認識の難しさは、入力画像が上図に示されているように入力信号の数字の集まりでしか無いことです。このようなデータを何度も経験することでネコを識別できるようにする必要があります。コンピュータに入力される画像は数字の塊に過ぎません。

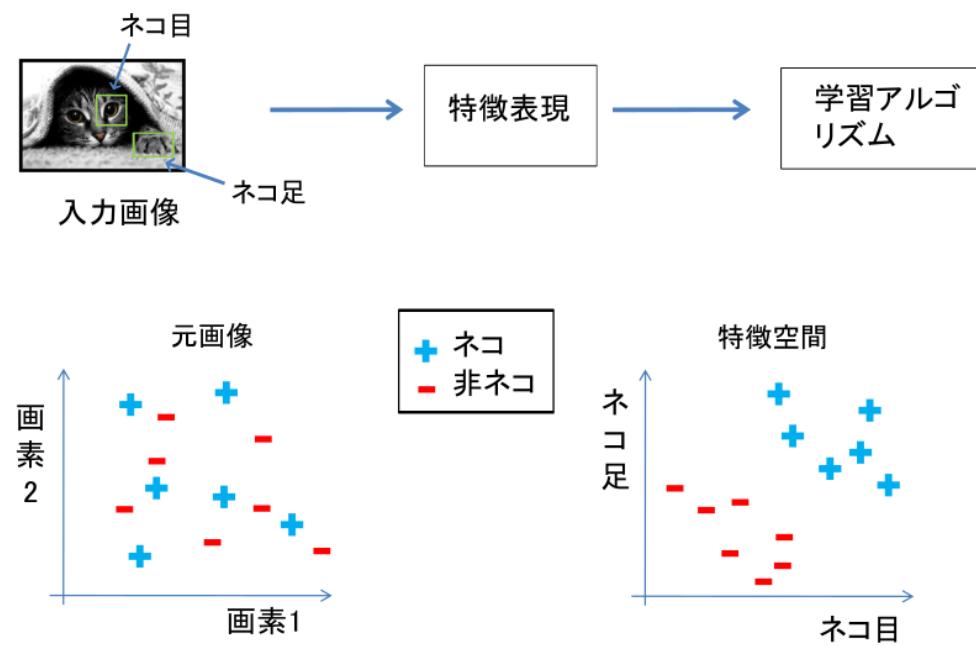
状況ごとにとるべき操作を命令として逐一コンピュータに与える指示する手順の集まりのことをコンピュータプログラムと呼びます。人間がコンピュータに与えることができる操作や命令によって画像認識システムを作る場合、命令そのものが膨大になったり、そもそも説明することが難しかったりします。例を挙げれば、お母さんを思い浮かべてくださいと言われば誰でも、それぞれ異なるイメージであれ思い浮かべることができます。また、提示された画像が自分の母親のものであるか、別の女性であるかの判断は人間であれば簡単です。ところがコンピュータには難しい課題となります。加えて母親の特徴をコンピュータに理解できる命令としてプログラムすることも難しい課題です。つまり自分の母親の特徴を曖昧な言葉でなく明確に説明するとともに難しい課題となります。というのは、女性の顔写真であればどの写真も似ていると言えるからです。顔の造形や輪郭、髪の位置などはどの画像も類似していることでしょう。ところがコンピュータにはこの似ている、似ていないの区別が難しいのです。

加えて、同一ネコの画像であっても、被写体の向き視線の方向や光源の位置や撮影条件が異なれば画像としては異なります。下図に示したように入力画像の中の特定の値だけを調べてみても、入力画像がネコである、そうではないかを判断することは難しい課題になります。

機械学習による特徴表象

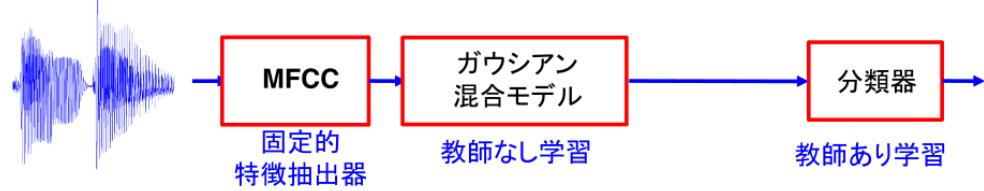


現在の画像認識では、特定の画素の情報に依存せずに、入力画像が持っている特徴をとらえるように設計されます。たとえば、ネコを認識するために必要ことは、ネコに特徴的な「ネコ目」や「ネコ足」を検出することであると考えます。入力画像から、ネコの持つ特徴を抽出することができれば、それらの特徴を持っている入力画像はネコであると判断して良いことになります(下図)。

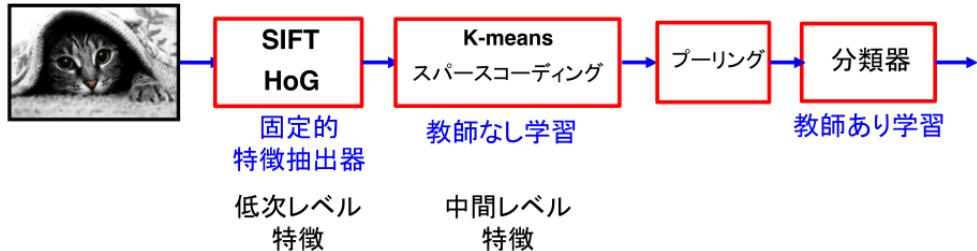


下図は、音声認識と画像認識の両分野において CNN が用いられる以前の従来手法をまとめたものです。

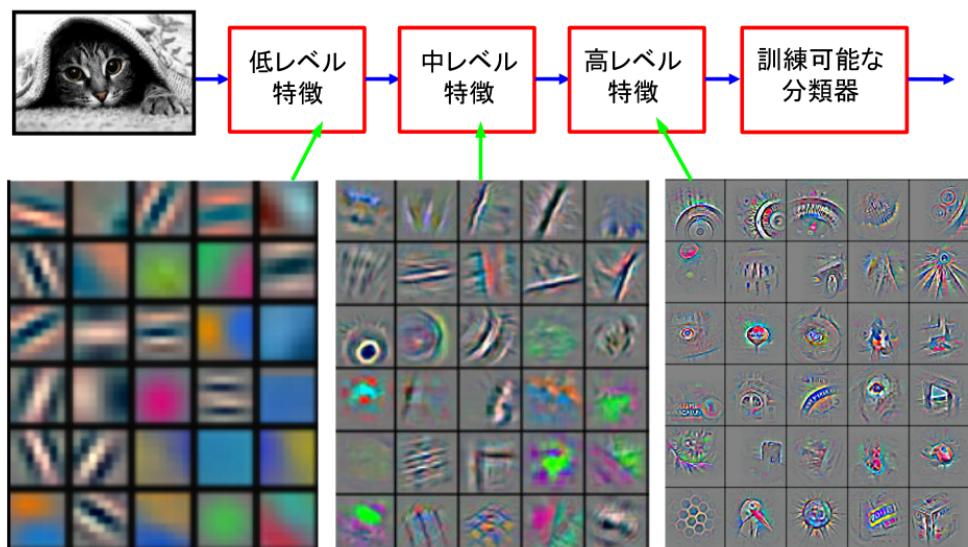
1990年代から2011年までの音声認識



2006年から2011年までの物体認識



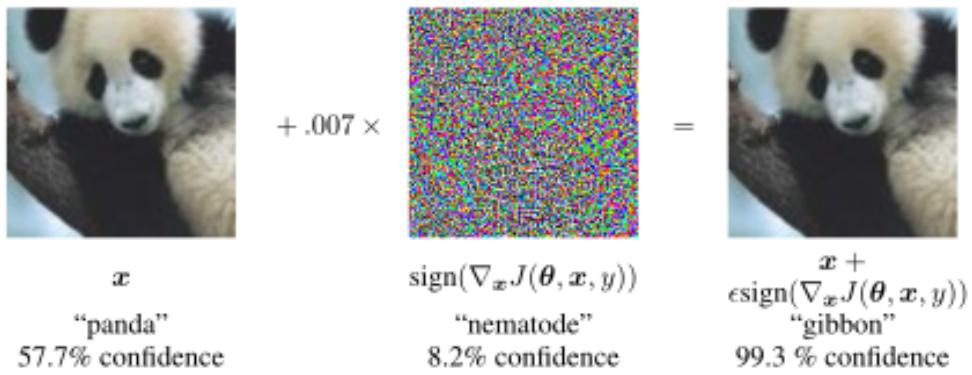
上図のような従来手法に対して、CNNではエンドツーエンドな特徴抽出を多層多段に重ねることによって複雑な特徴を抽出(検出)しようとしています(下図)。



コンピュータにはネコ目特徴検出器、ネコ足特徴検出器は備わっていません。そこで画像認識研究では、画像の統計的性質に基づいて特徴検出器を算出する方法を探る努力が行われてきました。しかし、コンピュータにネコ目特徴やネコ足特徴を教えるは容易なことではありません。このことは画像処理の分野だけに限りません、音声認識でも言語情報処理でもそれぞれの特徴器を一つ一つ定義し、チューニングするのは時間がかかり、専門的な知識も必要で困難な作業でした。

まとめると、1950年代後半以来:固定的、手芸的特徴抽出器と学習可能な分類器を用いた認識システムを作ることが試みられてきたといえます。これに対してCNNが主流となった現在はエンドツーエンドで学習可能な特徴抽出器を多数重ね合わせることで性能が向上しました。

夢のような話が続きましたが、本節の最後に逆にCNNは簡単に騙すことができる例を挙げておきます。



図では、左の画像が入力画像で、CNNは確信度57.7パーセントでパンダである認識しました。ところがこの画像に0.007だけ意味のない画像(図中央)を加えるた画像(図右)をCNNは99.3パーセントの確信度でテナガザル(gibbon)と判断しました。この例はここでは詳しく触れることはしませんが**敵対的学習**と呼ぶ訓練手法を説明する際に用いられた例です。

この例からも分かることは以下のようにまとめられるでしょう。すなわち、人間の脳を模したニューラルネットワークであるCNNが大規模化像認識チャレンジにおいて人間の認識性能を越えたと報道されました。ですが、人間の視覚認識を完全に実現したと考えるのは早計で、解くべき課題は未だ多数あるということです。この状況は、音声認識や言語情報処理でも同様であると言えます。

- ドロップアウト、データ拡張、各種正規化:cnn.md
- 有名なモデル LeNet, Alex Net, Inception, VGG, ResNet
- R-CNN, ハイウェイネット, YOLO, SSD
- セマンティックセグメンテーション
- 転移学習、事前学習、ファインチューニング

活性化関数 activation functions

```

import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

## [Original](https://github.com/alrojo/tensorflow-tutorial/\`\
## blob/master/lab1_FFN/lab1_FFN.ipynb)

# PLOT OF DIFFERENT OUTPUT USNITS
x = np.linspace(-6, 6, 100)
relu = lambda x: np.maximum(0, x)
leaky_relu = lambda x: np.maximum(0, x) + 0.1*np.minimum(0, x)
elu = lambda x: (x > 0)*x + (1 - (x > 0))*(np.exp(x) - 1)
sigmoid = lambda x: (1+np.exp(-x))**(-1)
def softmax(w, t = 1.0):
    e = np.exp(w)
    dist = e / np.sum(e)
    return dist
x_softmax = softmax(x)

plt.figure(figsize=(6,6))
plt.plot(x, relu(x), label='ReLU', lw=2)
plt.plot(x, leaky_relu(x), label='Leaky ReLU', lw=2)
plt.plot(x, elu(x), label='Elu', lw=2)
plt.plot(x, sigmoid(x), label='Sigmoid', lw=2)
plt.legend(loc=2, fontsize=16)
plt.title('Non-linearities', fontsize=20)
plt.ylim([-2, 5])
plt.xlim([-6, 6])

# softmax
# assert that all class probabilities sum to one
print(np.sum(x_softmax))
assert abs(1.0 - x_softmax.sum()) < 1e-8

```

[デモファイル 2019si_activation_functions.ipynb](#)

最終層（あるいは最終2層）は全結合層

ドット積を実行し、得られた結果に非線形活性化関数により出力を計算
ネットワーク全体は、一方の生画像ピクセルから他方のクラススコアまで、単一の微分可能なスコア関数を依然として表現している。最上位層の全結合層には損失関数（SVM/Softmax）がある。通常のニューラルネットワークの学習と同じ手法を用いる

CNN の詳細

通常のニューラルネットワークでは、直下層のニューロンとそのすぐ上の層の全ニューロンと結合を有する。一方 CNN ではその結合が部分的である。各ニューロンは多入力一出力の信号変換機とみなすことができ、活性化関数に非線形な関数を用いる点は通常のニューラルネットワークと同様。

画像処理を考える場合、典型的には一枚の入力静止画画像は3次元データである。次元は幅w、高さh、奥行きdであり、入力画像では奥行きが3次元、すなわち赤緑青の三原色。出力ニューロンへの入力は局所結合から小領域に限局される。

1. CNNの構成

CNN は以下のいずれかの層から構成される：

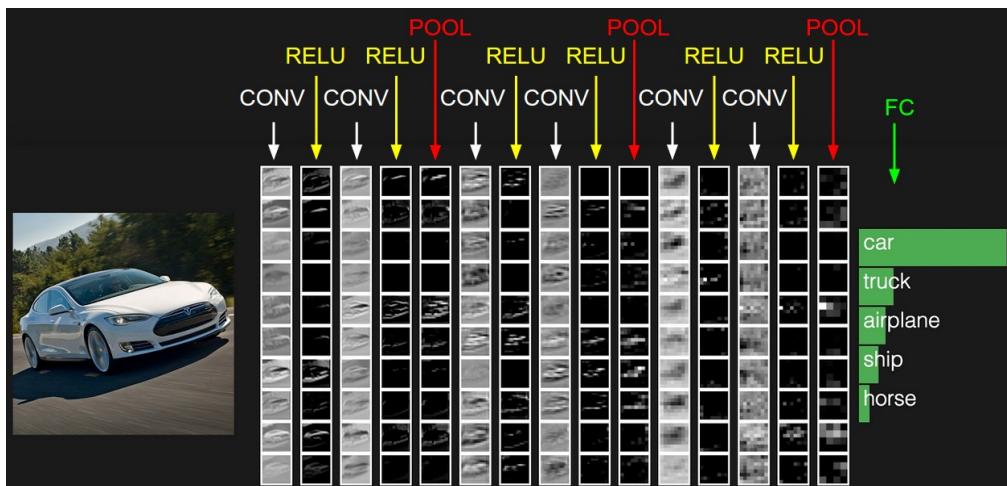
1. 疊込み層
2. プーリング層

3. 完全結合層（通常のニューラルネットワークと正確に同じもの、CNNでは最終1層または最終1,2層に用いる）

入力信号はパラメータの値が異なる活性化関数によって非線形変換される。畳込み層とプーリング層と複数積み重ねることで多層化を実現し、深層ニューラルネットワークとなる。

例：

- ・画像データを出力信号へ変換
- ・各層は別々の役割（畳込み、全結合、ReLU、プーリング）
- ・入力信号は3次元データ、出力信号も3次元データ
- ・学習すべきパラメータを持つ層は畳込み層、全結合層
- ・学習すべきパラメータを持たない層はReLU層とプーリング層
- ・ハイパーパラメータを持つ層は畳込み層、全結合層、プーリング層
- ・ハイパーパラメータを持たない層はReLU層



CNN アーキテクチャ：入力層は生画像の画素値(左)を格納、最後層は分類確率(右)を出力。処理経路に沿った活性の各ボリュームは列として示されている。3Dボリュームを視覚化することは難しかため、各ボリュームのスライスを行ごとに配置してある。最終層のボリュームは各クラスのスコアを保持するが、ソートされた上位5スコアだけを視覚化し、それぞれのラベルを印刷してある。

- ・入力層[32x32x3]: 信号は画像の生データ（画素値）幅w(32)、高さh(32)、色チャネル3(R, G, B)
- ・畳込み層: 下位層の限局された小領域のニューロンの出力の荷重付き総和を計算(内積、ドット積)。12個のフィルタを使用すると[32x32x12]となる。
- ・ReLU層の活性化関数は ReLU (Rectified Linear Unit) $\max(0, x)$
- ・プーリング層: 空間次元（幅,高さ）に沿ってダウンサンプリングを実行。[16x16x12]のようになる。
- ・全結合層はクラスに属する確率を計算: 10の数字のそれぞれがCIFAR-10の10カテゴリーの分類確率に対応するサイズ[1x1x10]に変換。通常のニューラルネットワーク同様、全結合層のニューロンは前層の全ニューロンと結合する。

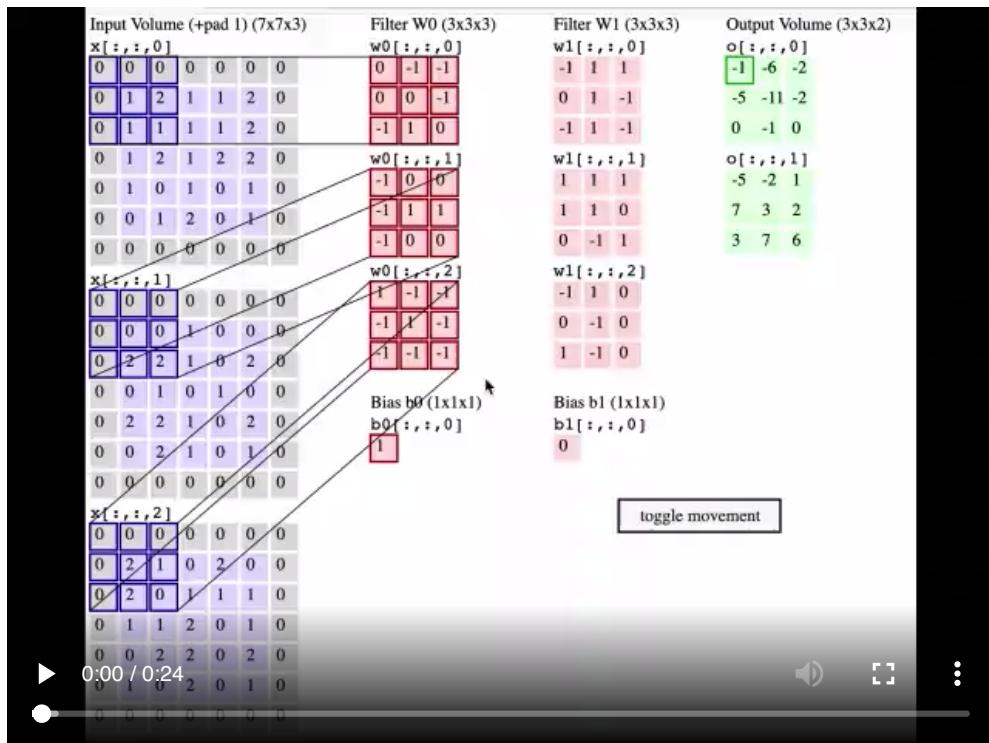
CNNは元画像（入力層）から分類確率（出力層）へ変換。学習すべきパラメータを持つ層（畳込み層、全結合層）とパラメータを持たない層（ReLU層）が存在。畳込み層と全結合層のパラメータは勾配降下法で訓練

2. 畳込み層

- ・畳込み層のパラメータは学習可能なフィルタの組
- ・全フィルタは空間的に（幅と高さに沿って）小さくなる
- ・フィルタは入力信号の深さと同一

- 第1層のフィルタサイズは例えば $5 \times 5 \times 3$ （5画素分の幅、高さ、と深さ3（3原色の色チャンネル））
- 各層の順方向の計算は入力信号の幅と高さに沿って各フィルタを水平または垂直方向へスライド
- フィルタの各値と入力信号の特定の位置の信号との内積（ドット積）。
- 入力信号に沿って水平、垂直方向にフィルタをスライド
- 各空間位置でフィルタの応答を定める2次元の活性化地図が生成される
- 学習の結果獲得されるフィルタの形状には、方位検出器、色プロップ、生理学的には視覚野のニューロンの応答特性に類似
- 上位層のフィルタには複雑な視覚パターンに対応する表象が獲得される
- 各畳込み層全体では学習すべき入力信号をすべて網羅するフィルタの集合が形成される
- 各フィルタは相異なる2次元の活性化地図を形成
- 各フィルタの応答特性とみなすことが可能な活性化地図
- フィルタの奥行き次元に沿って荷重総和を計算し、出力信号を生成

畳込み演算のデモ



局所結合: 画像のような高次元の入力を処理する場合、下位層の全ニューロンと上位層の全ニューロンとを接続することは**責任割当問題回避**の観点からもパラメータ数の増加は現実的ではない。代わりに各ニューロンを入力ボリュームのローカル領域のみに接続。空間的領域はニューロンの**受容野**と呼ばれるハイパー・パラメータ（フィルタサイズとも言う）。**深さ次元に沿った接続性=入力層の深さ次元**。空間次元（幅と高さ）と深さ次元をどのように扱うかにより、この非対称性を再び強調することが重要です。ニューロン間の結合は空間次元（幅と高さ）にそって限局的。入力次元の深さ全体を常にカバーする。

- 例1: 入力層のサイズが $[32 \times 32 \times 3]$ （RGB CIFAR-10画像データセットなど）であれば受容野（フィルタサイズ）が 5×5 とすれば、畳込み層内の各ニューロンは入力層の $[5 \times 5 \times 3]$ 小領域への結合係数を持つ。各小領域毎に $5 \times 5 \times 3 = 75$ の重み係数と1つのバイアス項が必要である。深さ次元に沿った上層のニューロンから下位層のニューロンへの結合は下位層の深さ(色チャンネル数)と等しく3である。
- 例2: 入力ボリュームのサイズが $[16 \times 16 \times 20]$ であるとすると 3×3 の受容野サイズで畳込層の全ニューロンの合計は $3 \times 3 \times 20 = 180$ 接続。接続性は空間的に局在する（ 3×3 ）が、入力深度（20）に沿っては完全結合

空間配置: 出力層ニューロンの数と配置については3つのハイパーパラメータで出力ニューロン数が定まる。

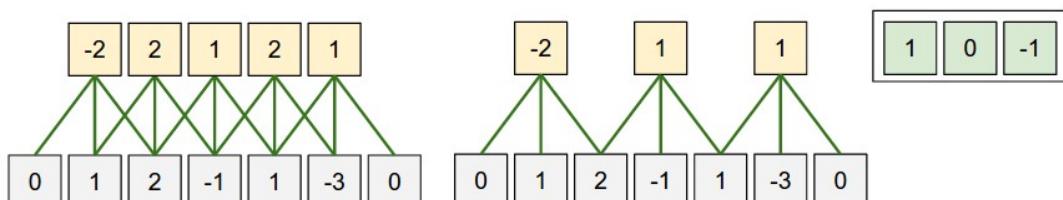
1. 深さ数(フィルタ数)

2. ストライド幅

3. ゼロパディング

4. 出力層ニューロン数のことを出力層の**深さ**数と呼ぶハイパーパラメータである。深さ数とはフィルタ数(カーネル数)とも呼ばれる。第1畳込み層が生画像であれば、奥行き次元を構成する各ニューロンによって種々の方位を持つ線分(エッジ検出細胞)や色ブロップのような特徴表現を獲得可能となる。入力の同じ領域を**深さ列**とするニューロン集団を**ファイバ**ともいう。
5. フィルタを上下左右にずらす幅を**ストライド幅**と呼ぶ。ストライド幅が1ならフィルタを1画素ずつ移動することを意味する。ストライドが2ならフィルタは一度に2画素ずつジャンプさせる。ストライド幅が大きければ入力信号のサンプリング間隔が大きく広がることを意味する。ストライド幅が大きくなれば上位層のニューロン数は減少する。
6. 入力の境界上の値をゼロで埋め込むことがある。これを**ゼロパディング**という。ゼロパディングの量はハイパーパラメータである。ゼロパディングにより出力層ニューロンの数を制御できる。下位層の空間情報を正確に保存するには入力と出力の幅、高さは同じである必要がある。

入力層のニューロン数を W 、上位にある畳込み層のニューロン数を F 、とすれば出力層に必要なニューロン数 S は、周辺のゼロパディングを P とすれば $(W - F + 2P)/S + 1$ で算出できる。たとえば下図でストライド1とゼロパディング0であれば入力7x7でフィルタサイズが3x3であれば $5 \times 5 = S = (7-3+2 \times 0)/1+1=5$ の出力である。ストライド2ならば $3 \times 3 = S = (7-3+2 \times 0)/2+1=3$ となる。



空間配置の例：入力空間の次元（x軸）が1つで受容野サイズ $F=3$ の場合、入力サイズ $W=5$ 、ゼロパディング $P=1$ であれば、

左図：出力層ニューロン数は $(5-3+2)/1+1=5$ の出力層ニューロン数となる。ストライド数 $S=1$ の場合。

右図： $s=2$ 、出力層ニューロン数 $(5-3+2)/2+1=3$ となる。ストライド $S=3$ ならばボリューム全体にきちんと収まらない場合もでてくる。数式で表現すれば $\lfloor((5-3+2)/4)\rfloor$ は3で割り切れないでの、整数の値として一意に決定はできない。

ニューロン結合係数は（右端に示されている）[1,0,-1]でありバイアスはゼロ。この重みはすべての黄色ニューロンで共有される。

ゼロパディング：上例では入力次元が5、出力次元が5であった。これは受容野が3でゼロ埋め込みを1としたためである。ゼロ埋め込みが使用されていない場合、出力ボリュームは、どれだけの数のニューロンが元の入力に「フィット」するのであろうかという理由で、空間次元がわずか3であったであろう。ストライドが $S = 1$ のとき、ゼロ埋め込みを $P = (F - 1)/2$ に設定すると、入力ボリュームと出力ボリュームが空間的に同じサイズになる。このようにゼロパディングを使用することは一般的である。CNNについて詳しく説明している完全な理由について説明する。

ストライドの制約: 空間配置ハイパーパラメータには相互の制約があることに注意。たとえば入力に $W = 10$ というサイズがあり、ゼロパディングは $P = 0$ ではなく、フィルタサイズは $F = 3$ 、 $(W - F + 2P)/S + 1 = (10 - 3 + 0)/2 + 1 = 4.5$ よりストライド $S = 2$ を使用することは不可能である。すなわち整数ではなくニューロンが入力にわたってきれいにかつ対称的に"適合"しないことを示す。

*AlexNet*の論文では、第一畳込層は受容野サイズ $F = 11$ 、ストライド $S = 4$ 、ゼロパディングなし $P = 0$ 。

畳込層 $K = 96$ の深さ $(227 - 11)/4 + 1 = 55$ 。畳込層の出力サイズは[55x55x96]。55x55x96ニューロンは入力領域[11x11x3]と連結。全深度列96個のニューロンは同じ入力領域[11x11x3]に繋がる。論文中には $(224-11)/4+1$ となっている。パディングについての記載はない。

パラメータ共有 パラメータ数を制御するために畳み込み層で使用される。上記の実世界の例を使用すると、最初の畳故意層には $55 \times 55 \times 96 = 290,400$ のニューロンがあり、それぞれ $11 \times 11 \times 3 = 363$ の重みと1のバイアスがある。これにより CNN 単独の第 1 層に最大 $290400 \times 364 = 105,705,600$ のパラメータが追加される。

パラメータ共有 により学習すべきパラメータ数が減少する。例えば [55x55x96] のフィルタでは深さ次元は 96 個のニューロンで、各深さで同じ結合係数を使うことにすればユニークな結合係数は計 $96 \times 11 \times 11 \times 3 = 34,848$ となるので総パラメータ数は 34,944 となる(バイアス項 +96)。各深さで全ニューロン(55x55)は同じパラメータを使用する。逆伝播での学習では、全ニューロンの全結合係数の勾配を計算する必要がある。各勾配は各深さごとに加算され 1 つの深さあたり一つの結合係数集合を用いる。

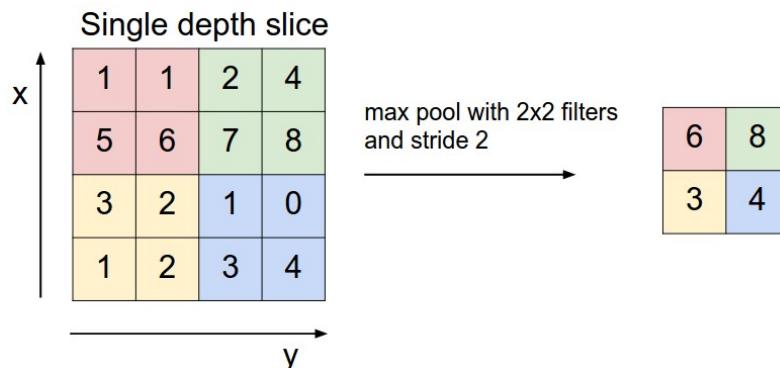
ある深さの全ニューロンが同じ重み係数ベクトルを共有する場合、畳み込み層の順方向パスは各深さスライス内で入力ボリュームとのニューロンの重みの **畳み込み** として計算できることに注意。結合荷重係数集合のことを **フィルタ** または **カーネル** と呼ぶ。入力信号との間で畳み込み演算を行うこととなる。



AlexNet の学習済フィルタ例：図の 96 個のフィルタは サイズ[11x11x3]。それぞれが 1 つの深さ内の 55×55 ニューロンで共有されている。画像の任意の位置で水平エッジ検出が必要な場合、画像の並進不变構造 translational-invariant structure 仮定により画像中の他の場所でも有効である。畳み込み層の出力ニューロン数は 55x55 個の異なる位置すべてで水平エッジの検出を再学習する必要はない。

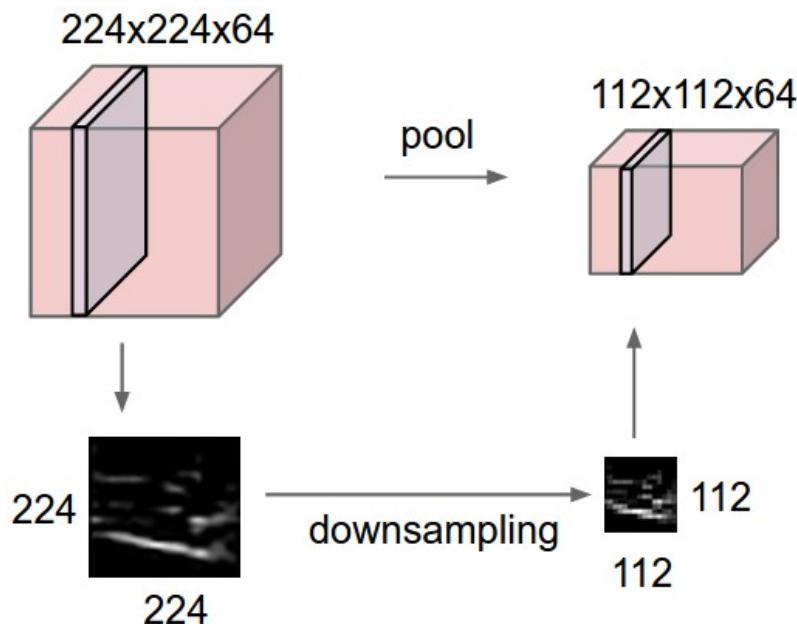
3. プーリング層

CNN では、連続する畳み込み層間にプーリング層を挿入するのが一般的。プーリング層の役割は、空間次元の大きさに減少させることである。パラメータ、すなわち計算量を減らし、過学習を制御できる。プーリング層は入力の各深さ毎に独立して動作する。最大値のみをとり他の値を捨てることを **マックスプーリング** と呼ぶ。サイズが 2x2 のフィルタによるプーリング層では、入力の深さごとに 2 つのダウンサンプルを適用し、幅と高さに沿って 2 ずつ増やして 75% の情報を破棄する。この場合 4 つの数値のうち最大値を採用することになる。



一般的なダウンサンプリング演算は **マックスプーリング** である。図ではストライド 2 すなわち 4 つの数値の中の最大値

平均プーリング. マックスプーリングではなく L_2 正則化プーリングを行う場合もある。平均プーリングは歴史的な意味があるがマックスプーリングの方が性能が良いとの報告がある。ある画像位置には物理的に一つの値だけが存在するという視覚情報処理が仮定すべき外界の物理的制約を反映していると文学的に解釈することも可能である。



プーリング層では、入力層ニューロン数の各深さについて空間的ダウンサンプリングを行う。この例はサイズ[$224 \times 224 \times 64$]の入力層ニューロン数がフィルタサイズ 2 でプールされ、サイズ 2 の出力ニューロン数 [$112 \times 112 \times 64$] は 2 倍である。奥行き数が保持されている。

4. 全結合層

全結合層のニューロンは、通常のニューラルネットワークと同じ
前層の全ニューロンと結合を持つ

5. CNN アーキテクチャ

1. 置込層
2. プーリング層
3. 全結合層

層は以上 3 種類が一般的。

6. CNN の層構造

入力層 → [[置込層 → ReLU] × N → プーリング(?)] × M → [全結合層 → ReLU] × K → 全結合層

最近のトレンドとしては大きなフィルタより小さなフィルタが好まれる傾向にある。

[3×3] が好まれる理由はど真ん中がある奇関数を暗黙に仮定しているためだと思われる（浅川の妄想）。その代わり多段にすれば [3×3] が 2 層で [5×5], 3 層で [7×7] の受容野を形成できるから受

容野の広さを層の深さとして実装しているとも解釈できる。1層で[7x7]の受容野より3層で[7x7]の受容野を実現した方がthe simpler, the betterの原則に沿っているとも（文学的）解釈が可能である（またしても浅川妄想）。

バックプロパゲーションの計算時に広い受容野を作るより層を分けた方がGPUのメモリに乗せやすいと言う計算上の利点もある。

リカレントニューラルネットワーク RNN

リカレントニューラルネットワーク（RNN）とは、時間的な変化や順序といった系列情報を扱うニューラルネットワークモデルです。このため音声認識、自然言語処理、ロボットの生成制御などに用いられています。時々刻々変化するデータを扱うには、それまでに処理されたデータの系列を文脈として保持しておく必要があります。リカレントニューラルネットワークを拡張した長短期記憶モデル(LSTM: Long short-term memory)を用いる場合が多いです。

応用事例としては、最近マイクロソフトが開発した女子高生人工知能「りんな」の対話生成アルゴリズムが有名です。LINEであたかも女子高生と会話しているかのようなコミュニケーションができることで話題になりました。

そのほかにも自動翻訳、画像と文章と相互変換（画像を入力するとその画像を説明する文章を生成する、逆にある文章を与えると対応する画像を生成する）などがあります。リカレントニューラルネットワークを用いた技術は他の従来手法の性能を上回り、現時点での最高性能と認められています。

現段階で、人間が書いたものなのかコンピュータが書いた文章なのか区別がつかない場合すらあります。新たなチューリングテスト（あるいはチューリングチャレンジ、コンピュータの生成した文か人間が書いた文かが見分けがつかなければ、もはやコンピュータに知性が宿っていると言っても良いだろうという考え方）と言ったりもします。

- 実習1 シーケンシャル kmnist
- 実習1 足し算 RNN
-

- 勾配消失問題、勾配爆発問題:
- 勾配チェック、勾配クリップ:
- 言語モデル:
- 埋め込みモデルによる意味表現:
- 潜在意味解析、トピックモデル(TDIDF, LSI, LSA, topic model):
- 注意(seq2seq, transformer):
- 機械翻訳、sentence vector (skip-thought, ELMo, BERT, Big bird, Tough-to-beat):
- 各種評価指標 sacre BLUE, negative log likelihood, perplexity:
- 画像脚注付け (NIC), pix2pix, img2txt, VQA:
- ニューラルチューリングマシン:

リカレントニューラルネットワークの特徴

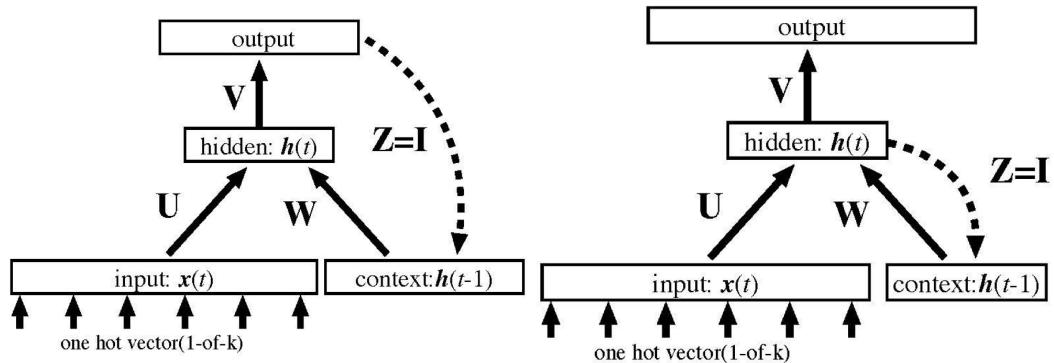
- 過去の状態を多様に保持する中間層
- 中間層更新は非線形
- 深層化（多層中間層）
- 駄菓子菓子、数多くの黒魔法が存在
 - 勾配クリップ、勾配正規化、勾配チェック、忘却バイアス、正規化、正則化、IRNN

--

最近の成果

2. 音声認識(Graves & Jaitly, 2014; Graves, Mohamed, & Hinton, 2013)
 3. 手書き文字生成(Graves, 2013)
 4. 系列学習(Sutskever, Vinyals, & Le, 2014)
 5. 機械翻訳(Bahdanau, Cho, & Bengio, 2015; Luong, Sutskever, Le, Vinyals, & Zaremba, 2015)
 6. 画像脚注付け(Kiros, Salakhutdinov, & Zemel, 2014; Vinyals, Toshev, Bengio, & Erhan, 2015)
 7. 構文解析(Vinyals et al., 2015)
 8. プログラムコード生成(Zaremba & Sutskever, 2015)
-

古典的リカレントニューラルネットワーク



ミコロフ革命

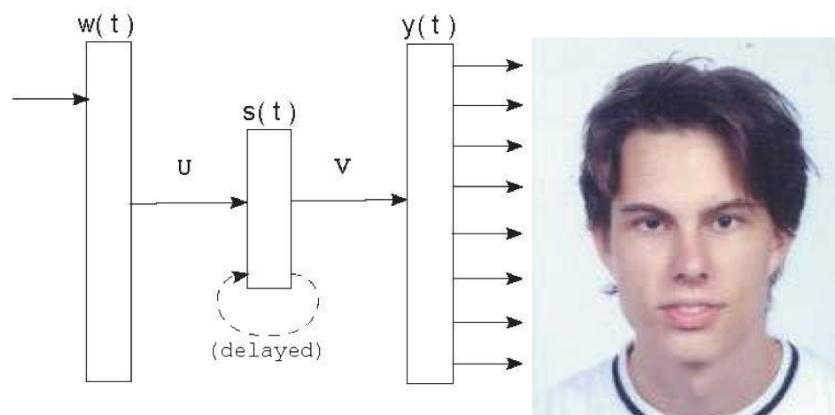


Fig. 1. Simple recurrent neural network.

$$\begin{aligned} s(t) &= f(\mathbf{U}w(t) + \mathbf{W}s(t-1)) \\ y(t) &= g(\mathbf{V}s(t-1)) \\ U(t+1) &= U(t) + \alpha w(t) e_h(t)^\top \end{aligned}$$

リカレントニューラルネットワークの仲間

- アトラクターネットワーク
- ホップフィールドネットワーク
- エコーステートネットワーク
- ボルツマンマシン（制限付きではない方）

BPTT, LSTM

How does LSTM work?

1. LSTM はロジスティックユニットやハイパータンジェントでアナログ値を保持している隠れ層のメモリユニットを置き換えたモデル
2. 各メモリセルは入出力ゲートを制御
3. メモリセルに保存されているアナログ値には忘却ゲートが付いている
4. 入出力ゲートが閉じていて忘却ゲートが開いていれば、次の時刻も状態を保持する Le, Jaitly, & Hinton (2015)

BPTT, LSTM 解題

単純再帰型ニューラルネットワークSRN

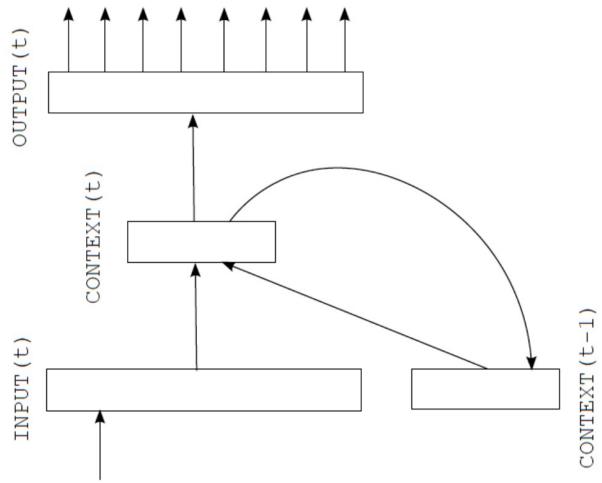
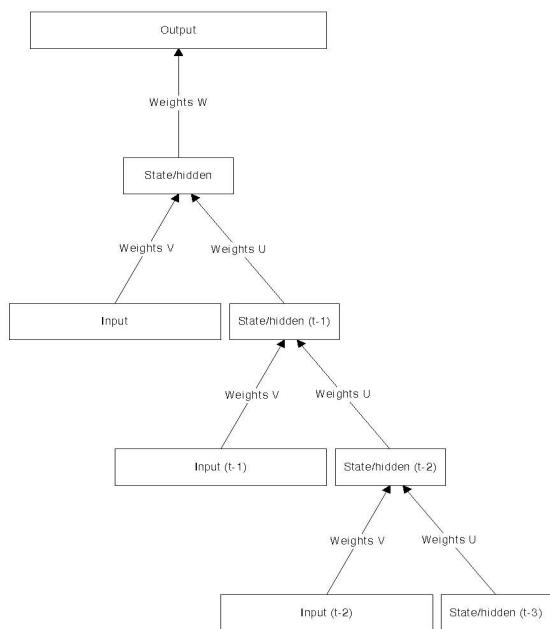


Figure 1: Simple recurrent neural network.

Mikolov (2010) Fig. 1

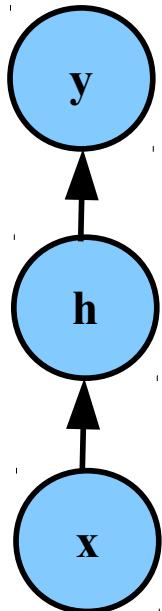
単純再帰型ニューラルネットワークSRN



Booden (2001) Fig. 5

Figure 5: The effect of unfolding a network for BPTT ($\tau = 3$).

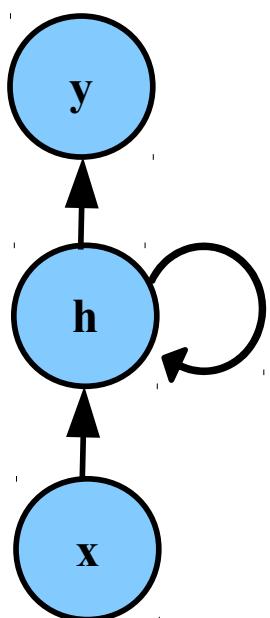
表記と基本グラフ



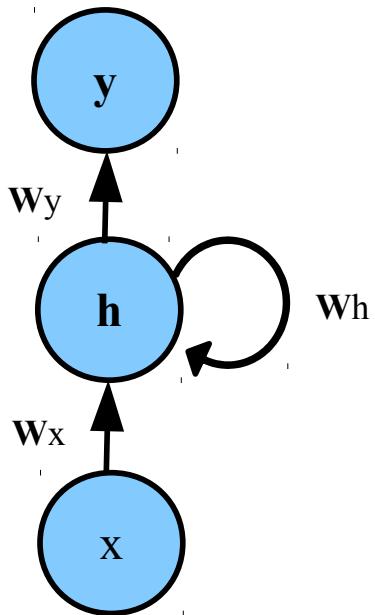
y: 出力層ニューロン

h: 中間層ニューロン

x: 入力層ニューロン



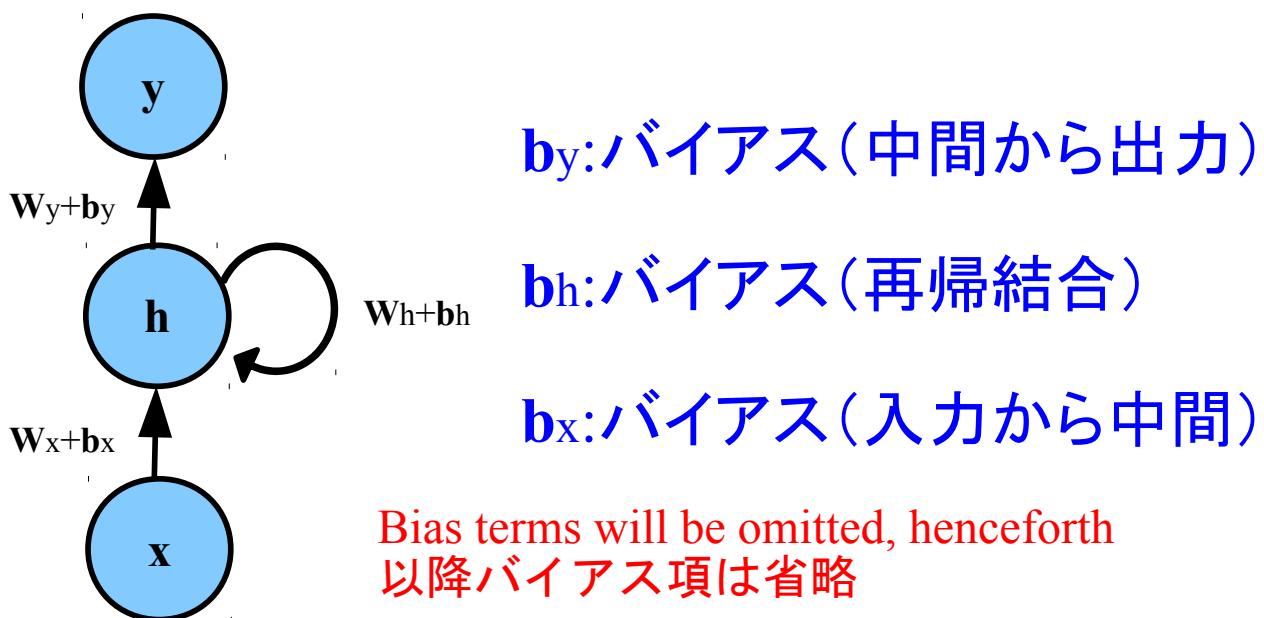
再帰結合 (recurrent connections)



W_y:結合係数行列(中間から出力)

W_h:結合係数行列(再帰結合)

W_x:結合係数行列(入力から中間)

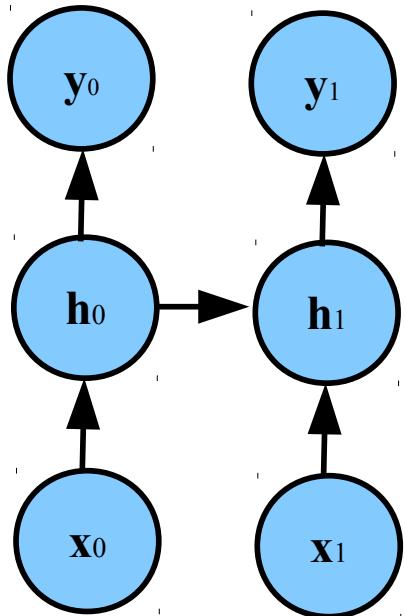


b_y:バイアス(中間から出力)

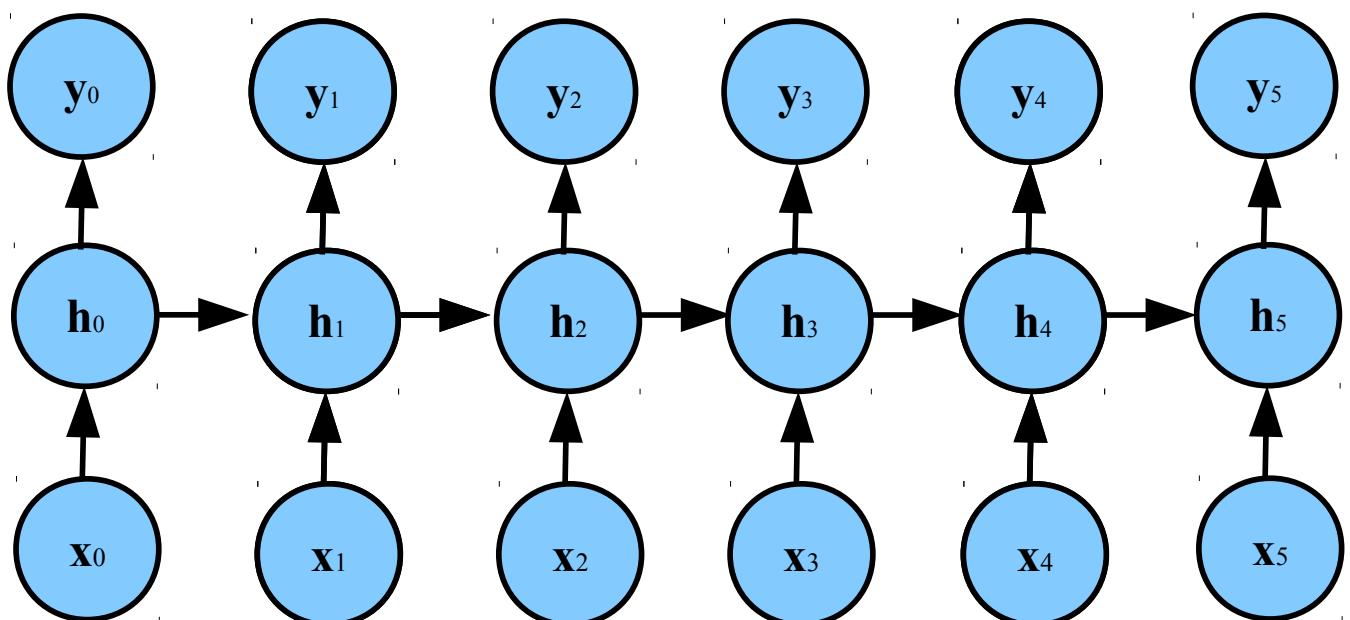
b_h:バイアス(再帰結合)

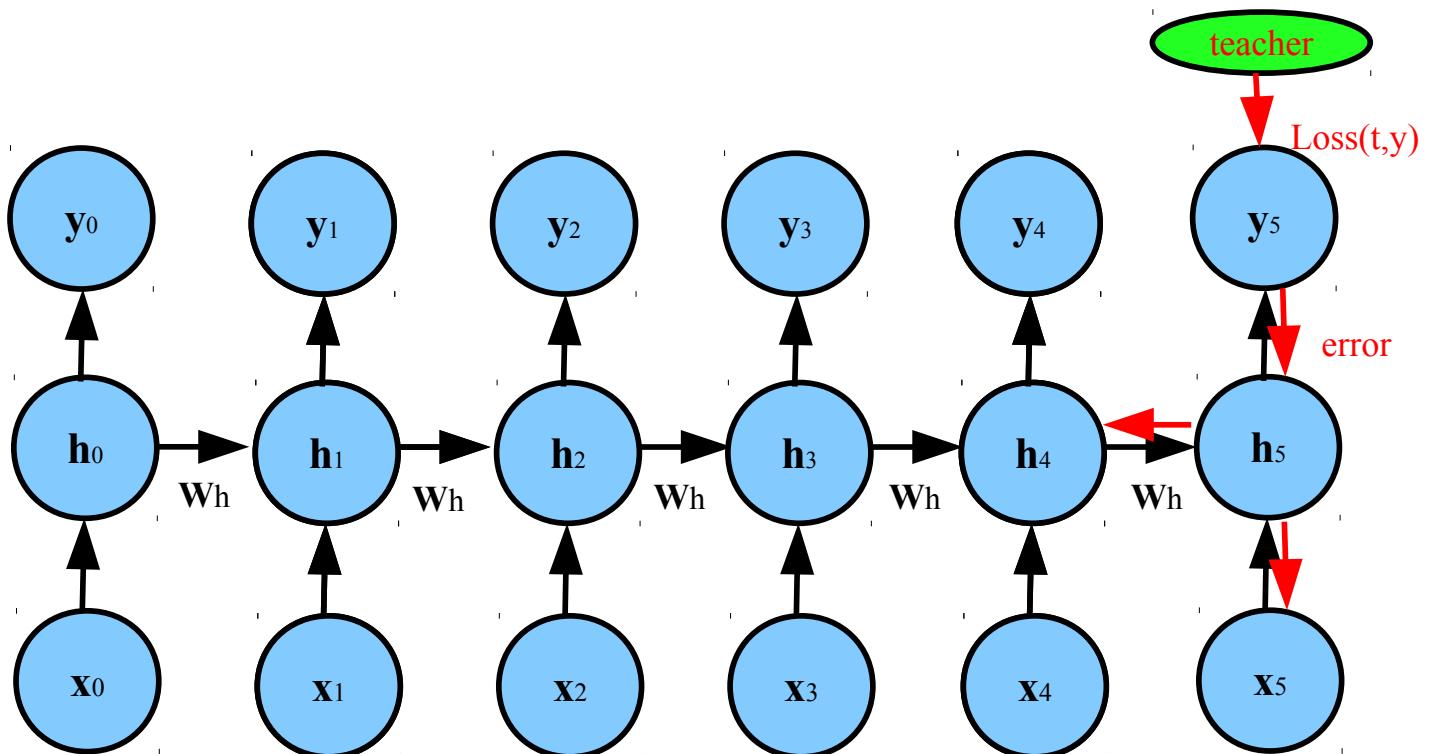
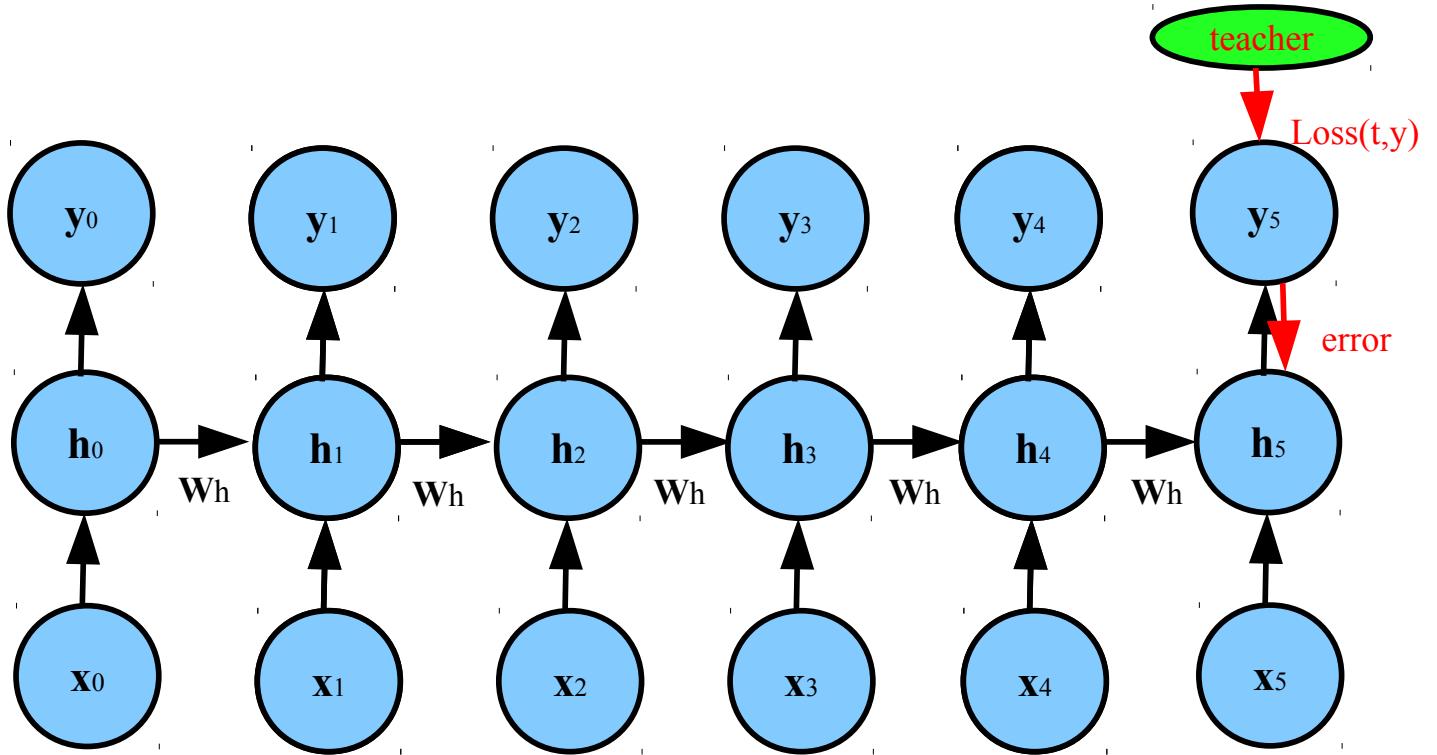
b_x:バイアス(入力から中間)

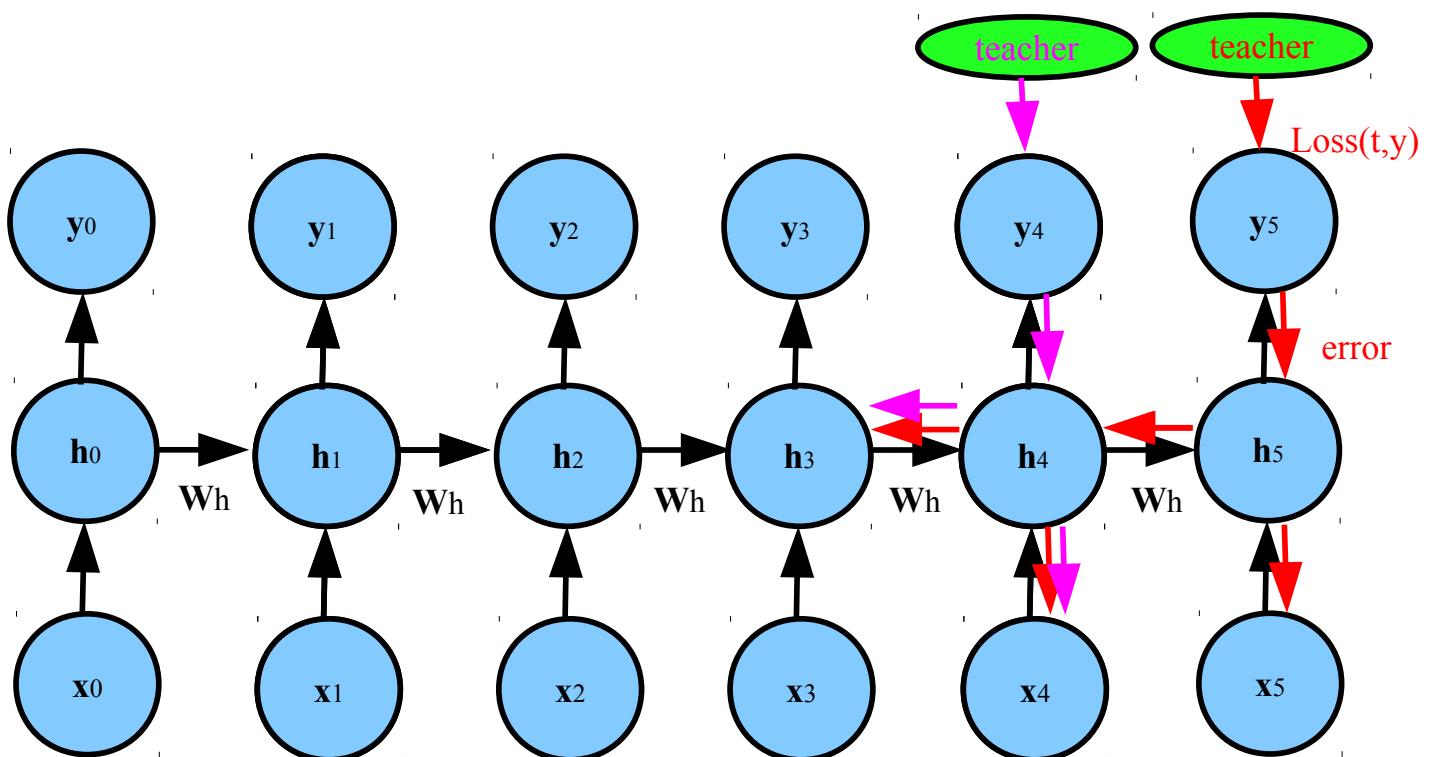
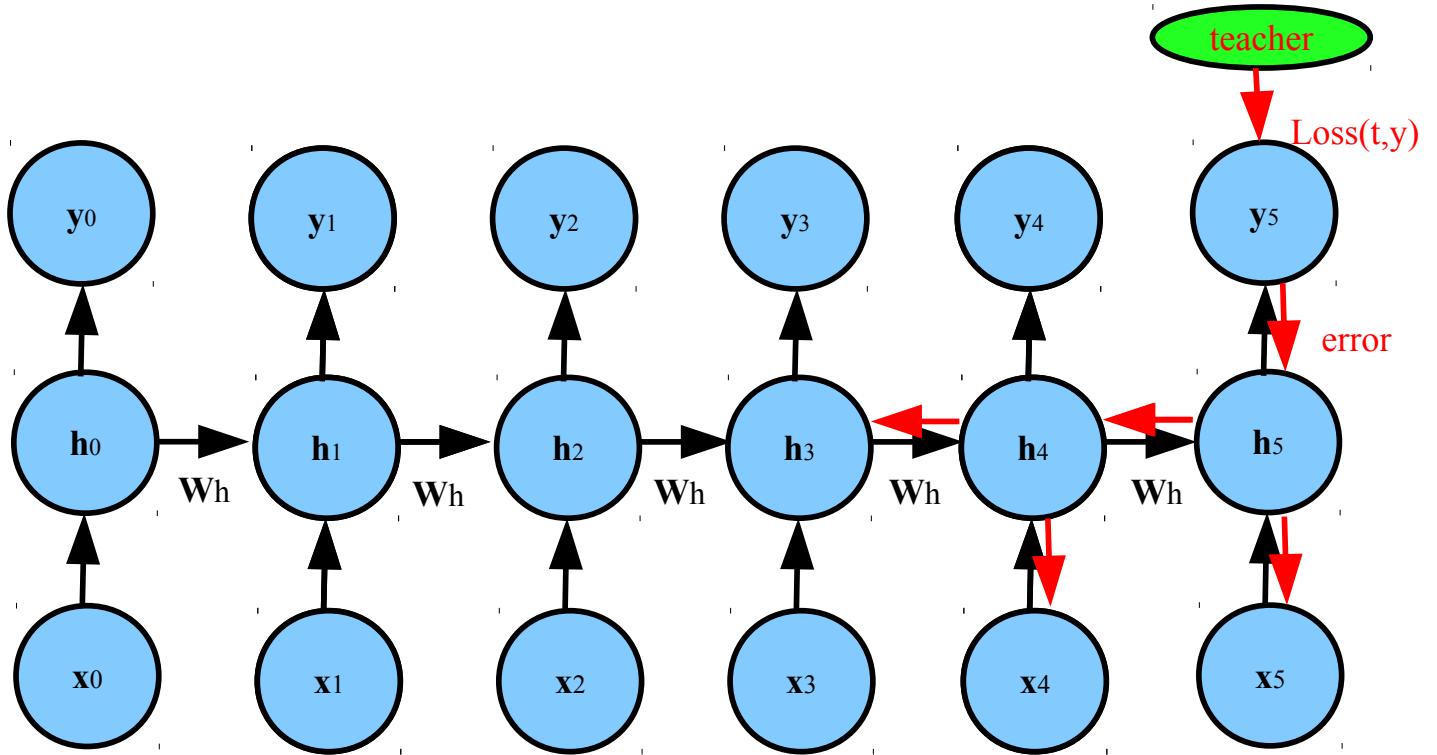
Bias terms will be omitted, henceforth
以降バイアス項は省略

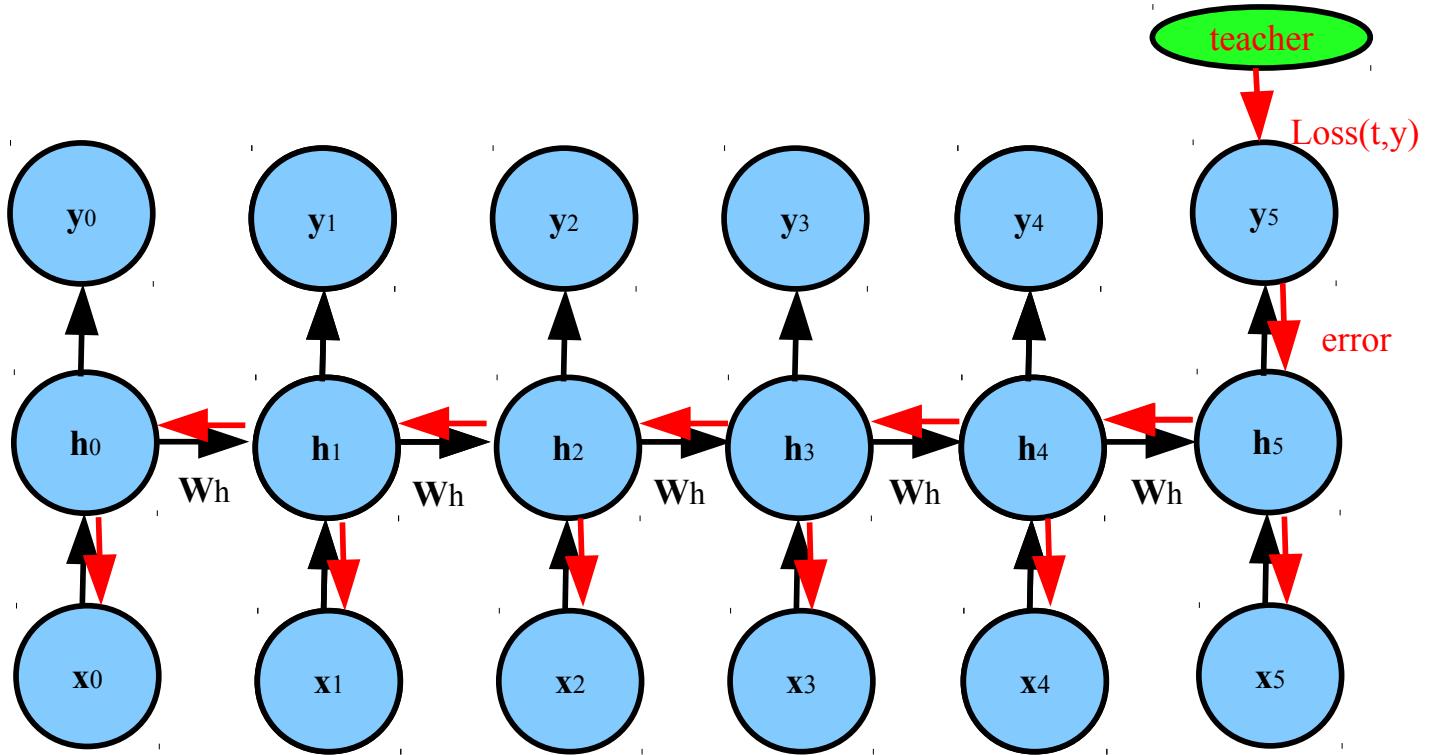


Digits subscripted indicate time
 $\tau := 0 \dots$
 下付き添字は時刻を表す。
 カッコで表記する流儀もある
 (e.g. $x(t)$)

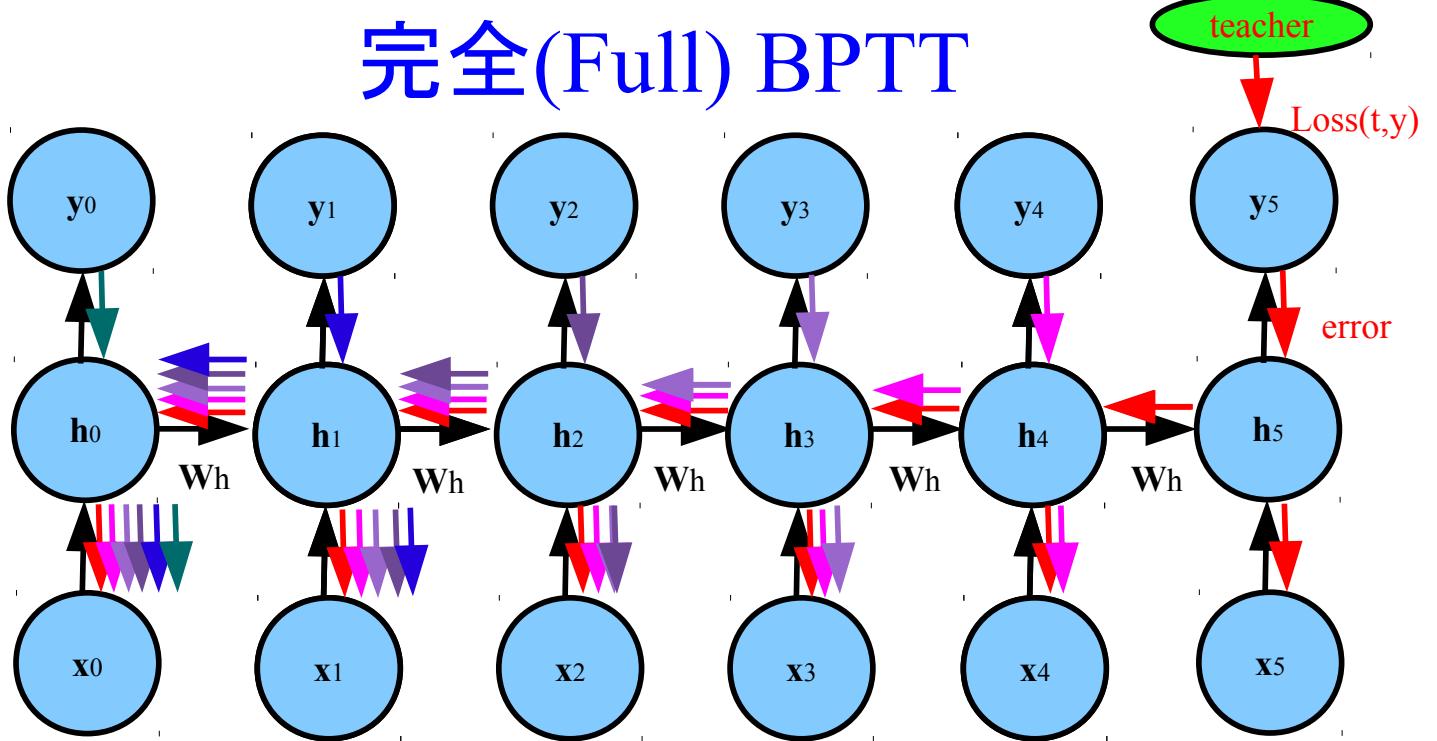




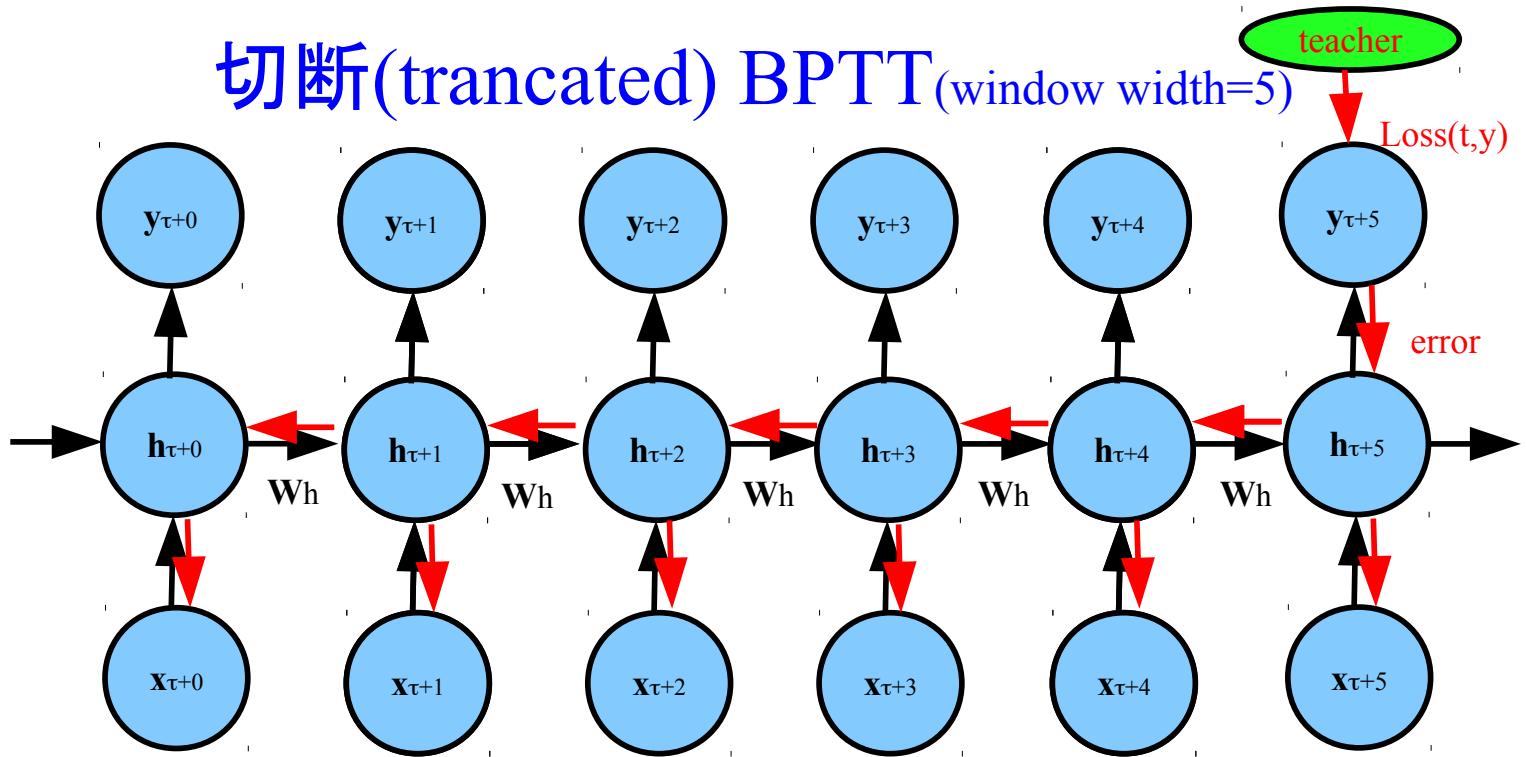




完全(Full) BPTT

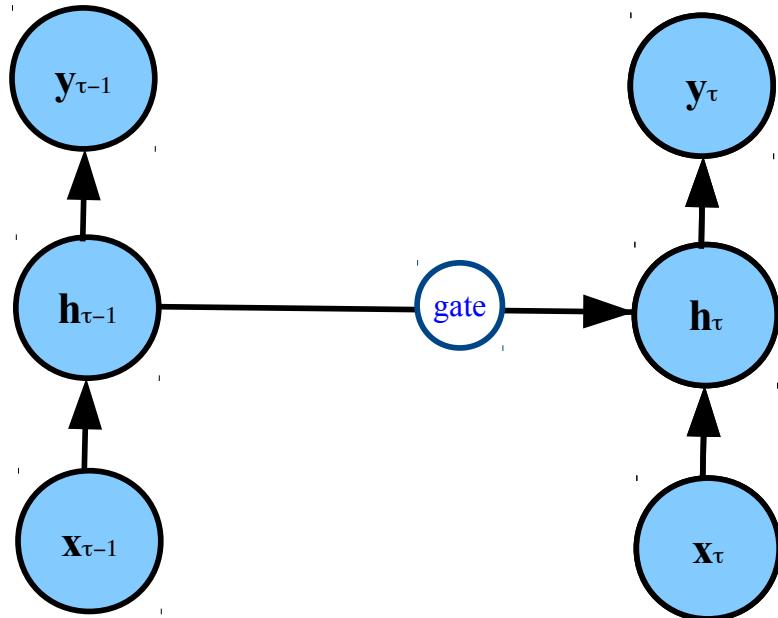


切断(trancated) BPTT(window width=5)

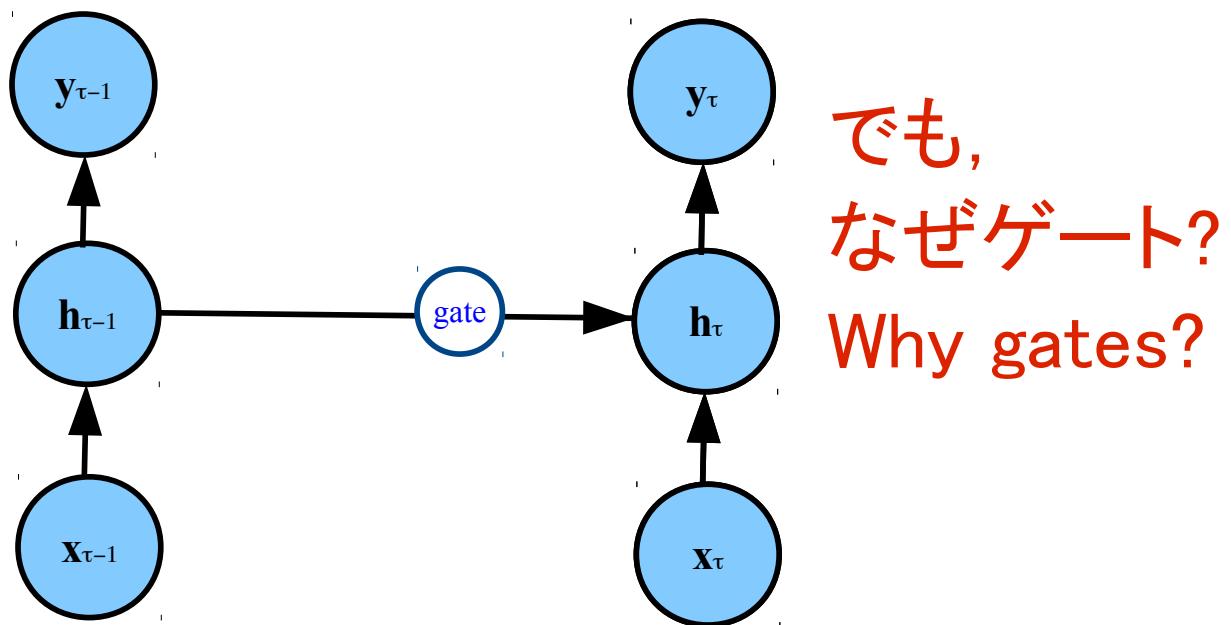


改良可能?
Can we improve?

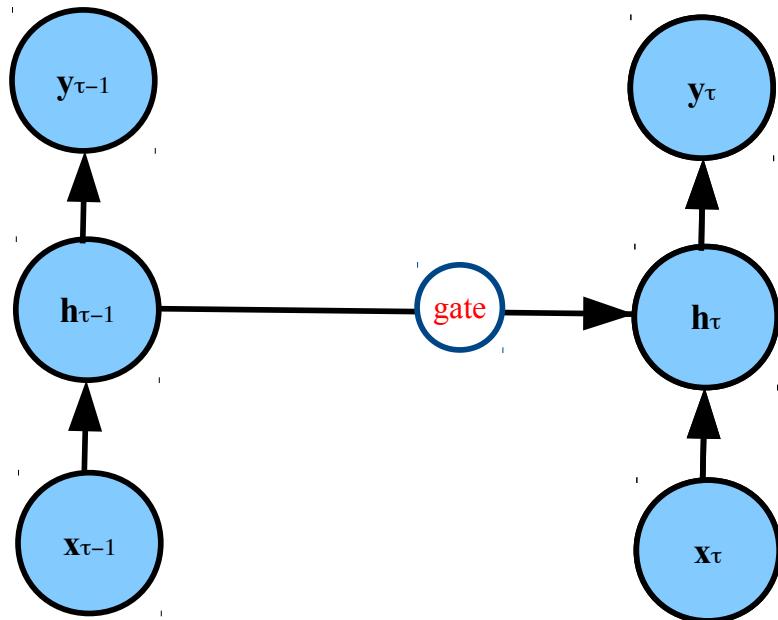
ゲートの導入 introducing gates to control hidden state



ゲートの導入 introducing gates to control hidden state

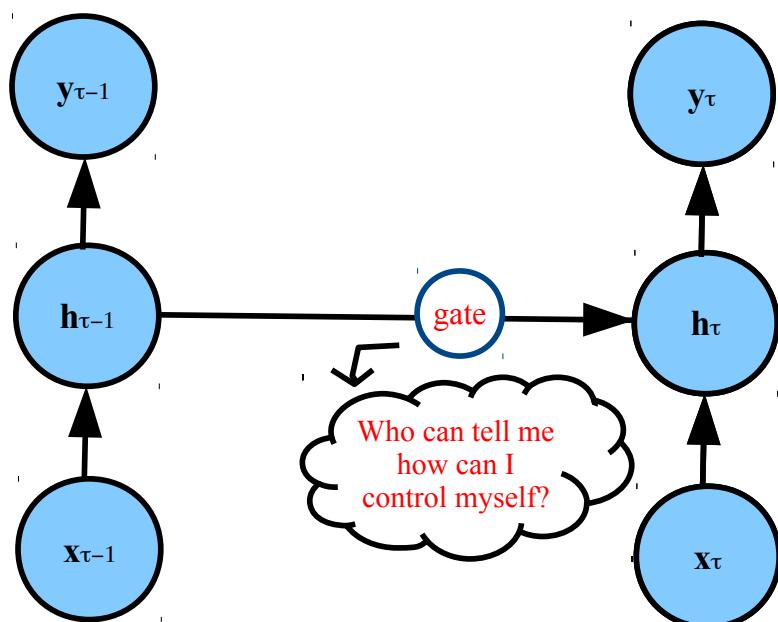


忘却ゲートの導入



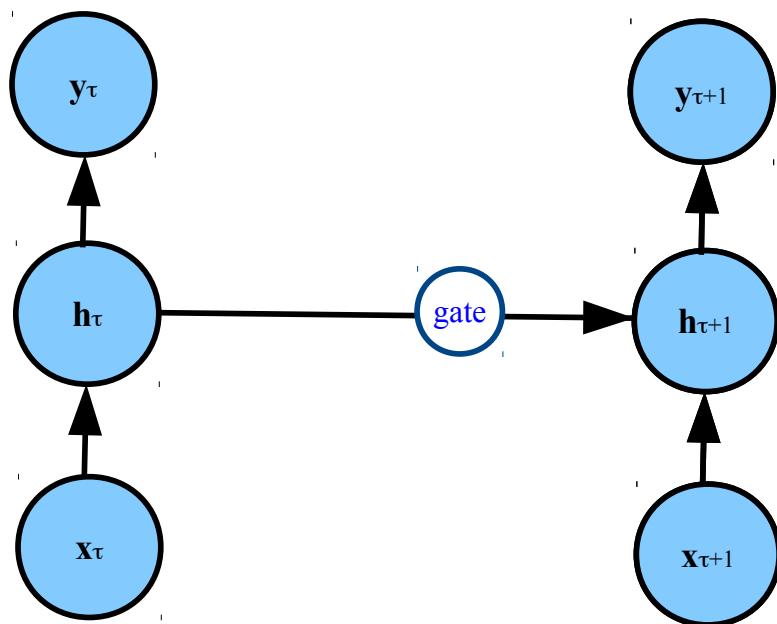
Who can control gates?
誰がどうやって
ゲート制御？

忘却ゲートの導入



Who can control gates?
誰がどうやって
ゲート制御？

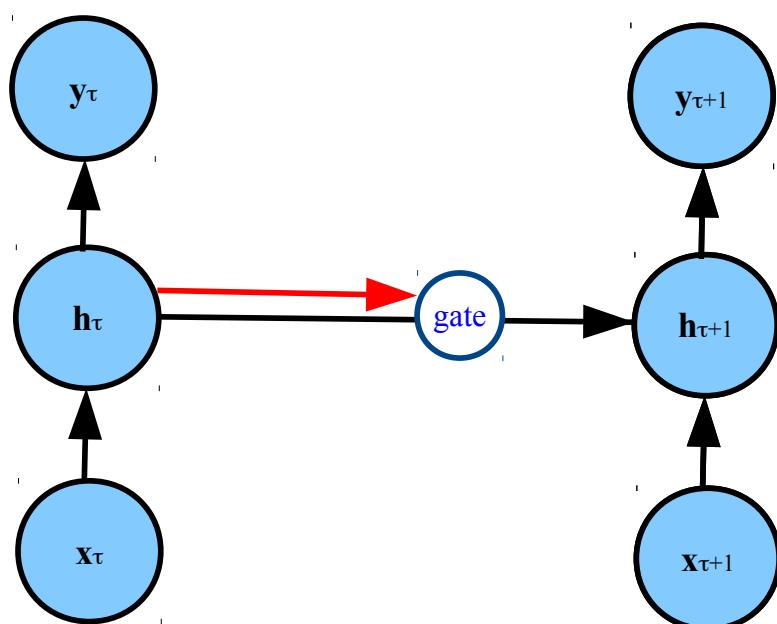
忘却ゲートの導入



who can control gates?
誰がどうやって
ゲートを制御？

3つ候補

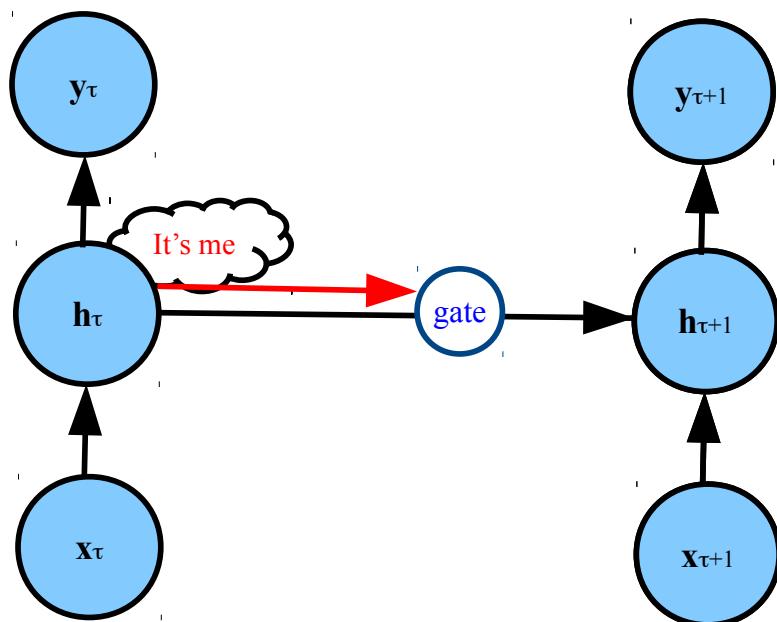
忘却ゲートの導入



who can control gates?
誰がどうやって
ゲートを制御？

3つ候補
1. h_τ

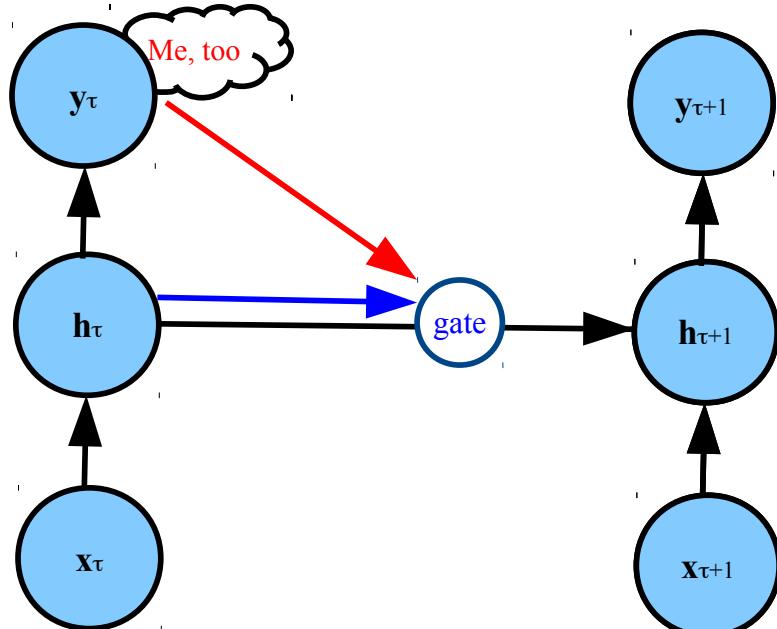
忘却ゲートの導入



who can control gates?
誰がどうやって
ゲートを制御？

- 3つ候補
1. h_τ

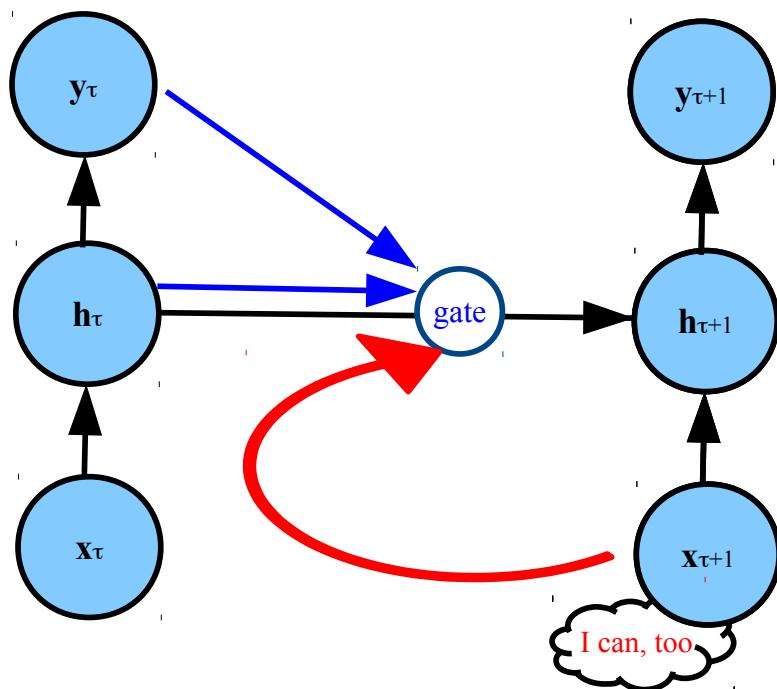
忘却ゲートの導入



who can control gates?
誰がどうやって
ゲートを制御？

- 3つ候補
1. h_τ
2. y_τ

忘却ゲートの導入



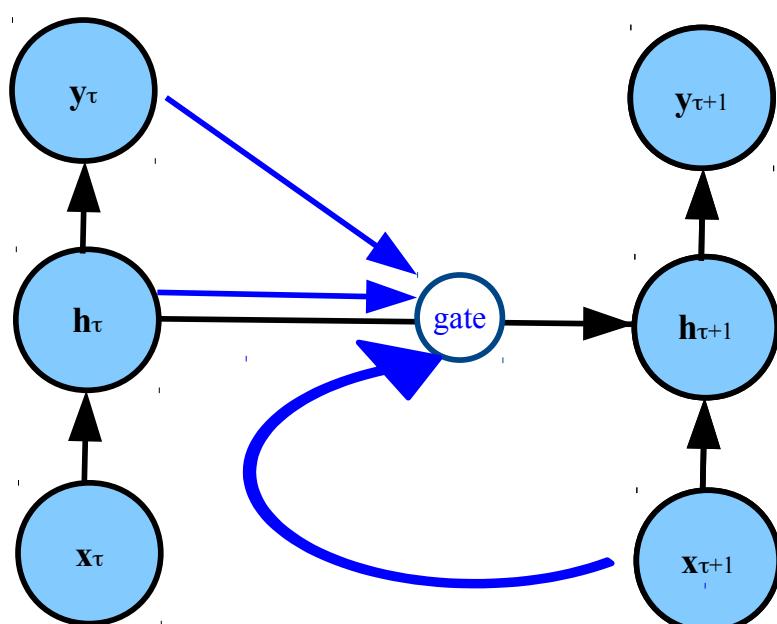
who can control gates?
誰がどうやって
ゲートを制御？

3つ候補

1. h_τ
2. y_τ
3. $X_{\tau+1}$

I can, too

忘却ゲートの導入

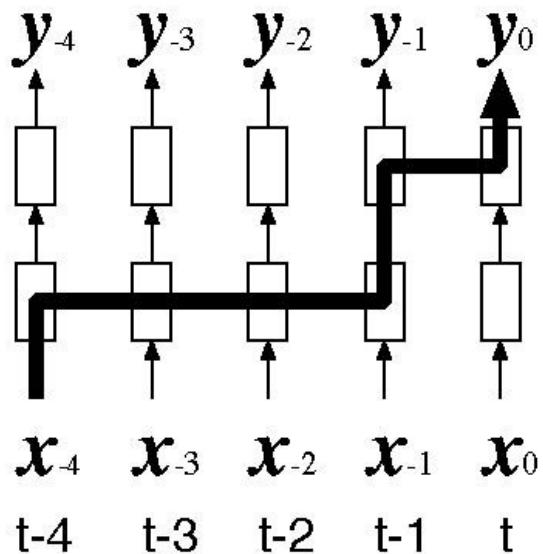


1. h_τ
2. y_τ ゲート制御
3. $X_{\tau+1}$

$$h_{\tau+1} = h_\tau \sigma(x)$$

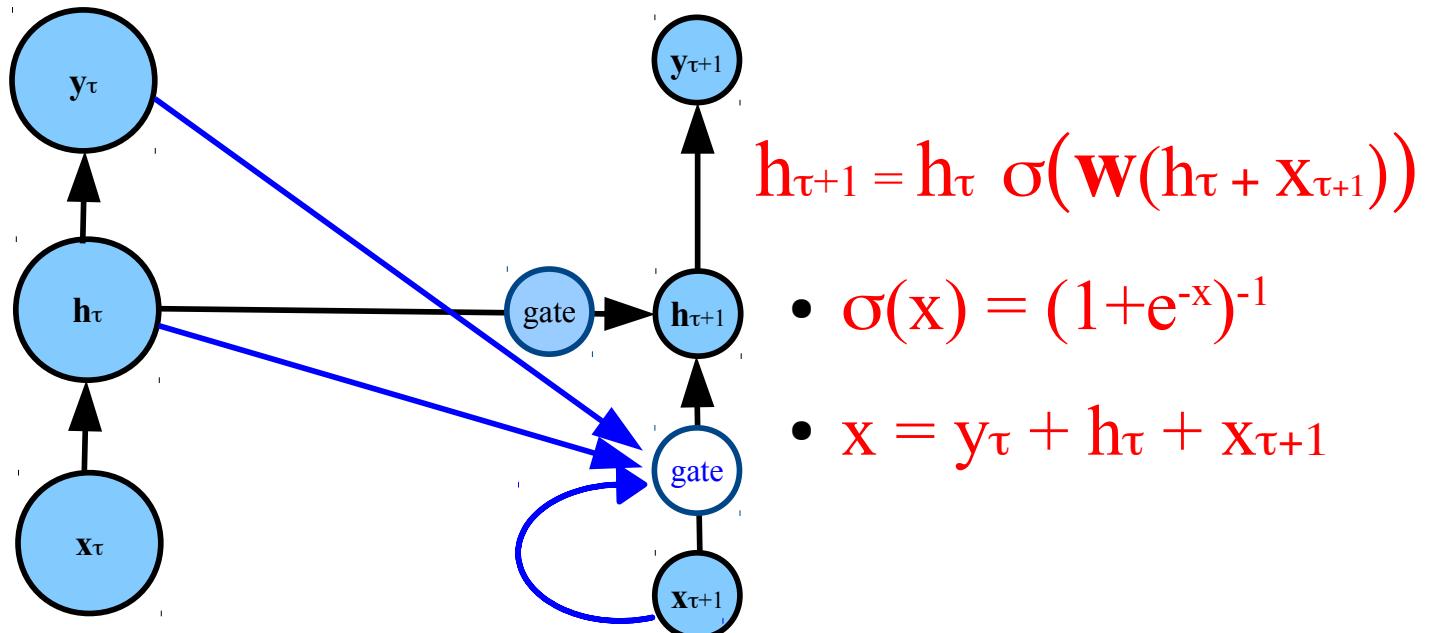
- $\sigma(x) = (1+e^{-x})^{-1}$
- $x = W_f(y_\tau + h_\tau + X_{\tau+1})$

ゲートによって長距離依存LTDを解消可能



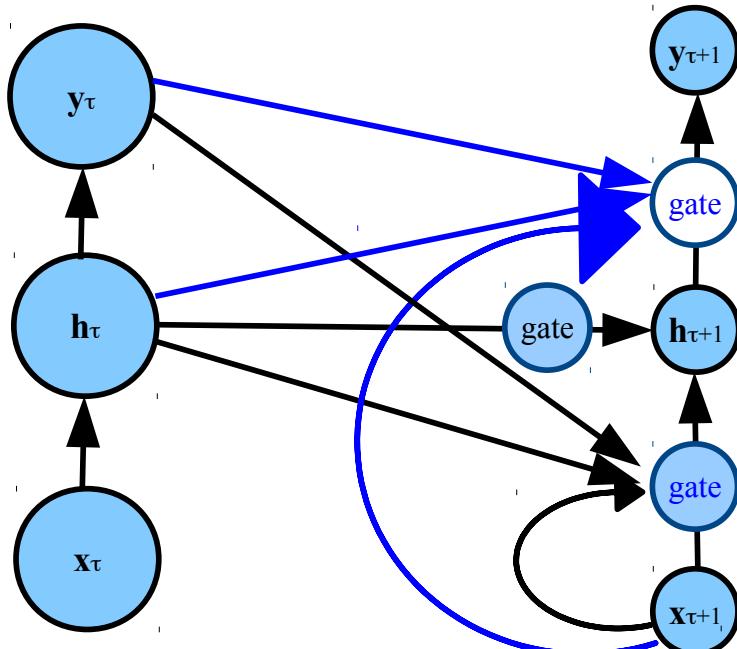
もっと改良可能？
Can we improve more?

入力ゲートの導入



もっともっと可能?
You need more?

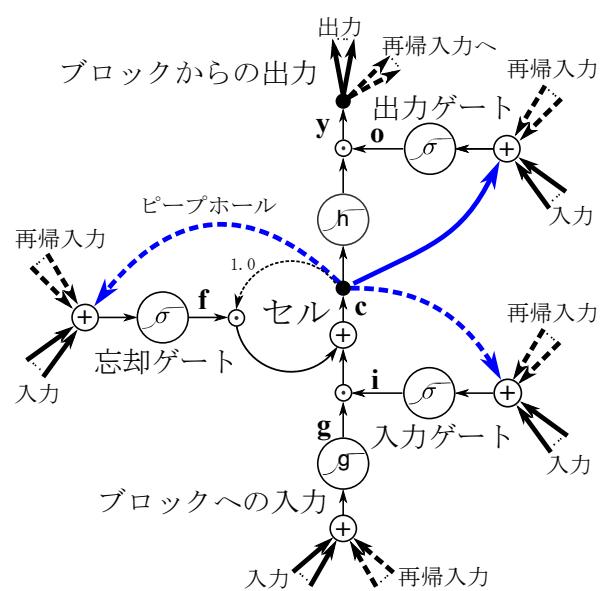
出力ゲートの導入



$$h_{\tau+1} = h_\tau \sigma(W(h_\tau + X_{\tau+1} + y_{\tau+1}))$$

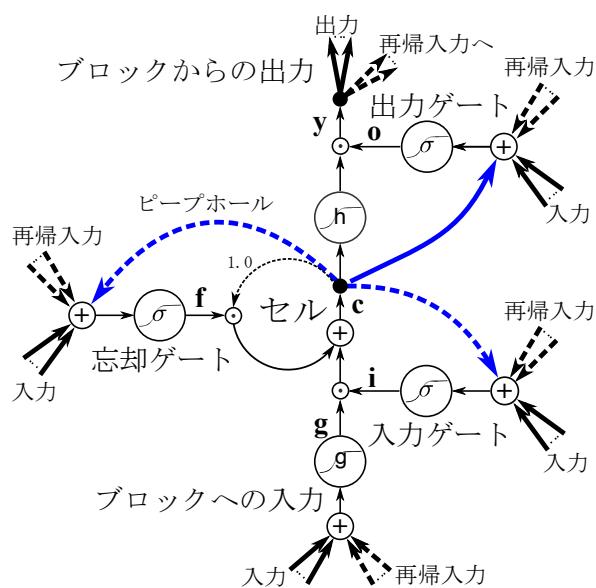
- $\sigma(x) = (1+e^{-x})^{-1}$
- $x = y_\tau + h_\tau + X_{\tau+1}$

LSTM



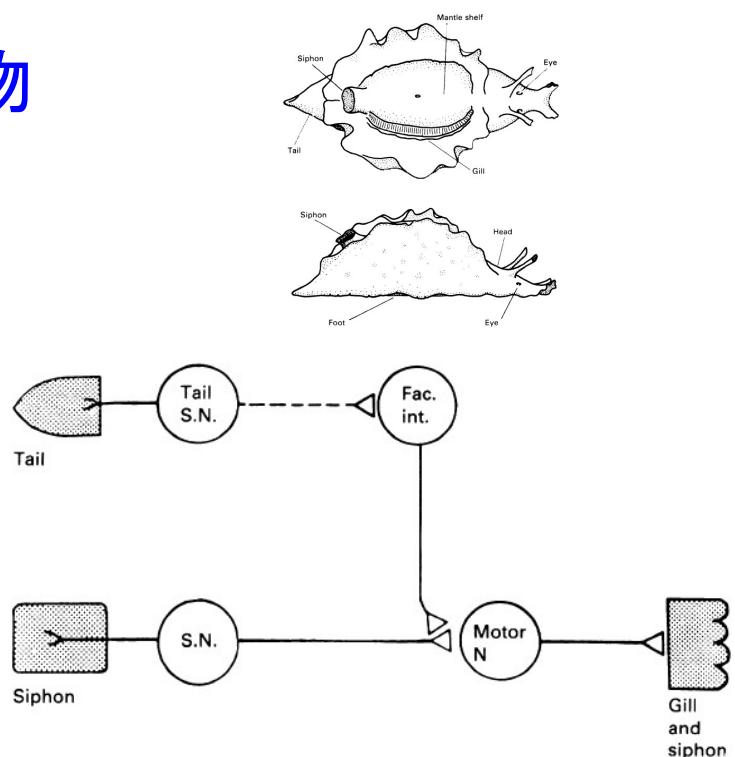
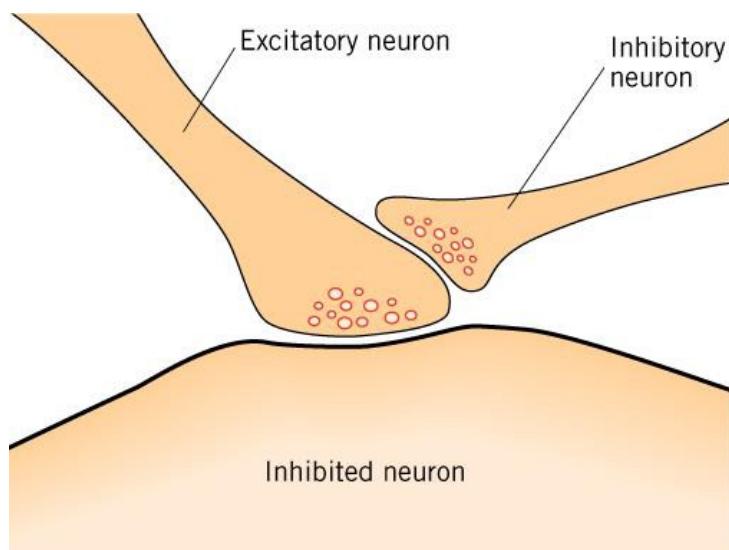
- 入力ゲート $i = \sigma(\text{)}$
- 忘却ゲート $f = \sigma(\text{)}$
- 出力ゲート $o = \sigma(\text{)}$
- 入力全体 $g = \phi(\text{)}$
- セル $c = f @ c + i @ g$
- 出力 $y = o @ \phi(\text{)}$

LSTM



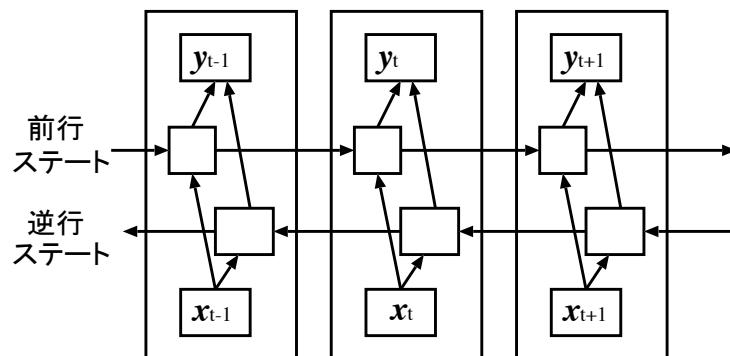
$$\begin{aligned}
 i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i), \\
 f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f), \\
 o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o), \\
 g_t &= \phi(W_{xc}x_t + W_{hc}h_{t-1} + b_c), \\
 c_t &= f_t \odot c_{t-1} + i_t \odot g_t, \\
 y_t &= o_t \odot \phi(c_t)
 \end{aligned}$$

LSTMの生理学的対応物



<http://kybele.psych.cornell.edu/~edelman/Psych-2140/week-2-2.html>

双方向 RNN



word2vec

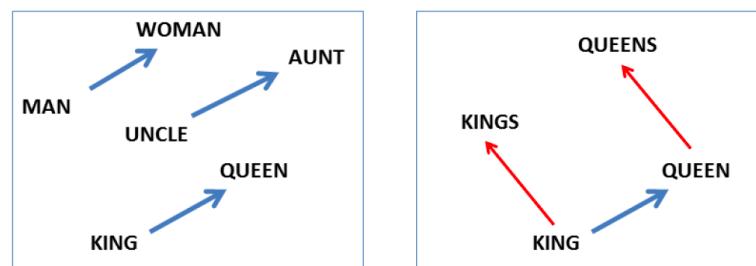
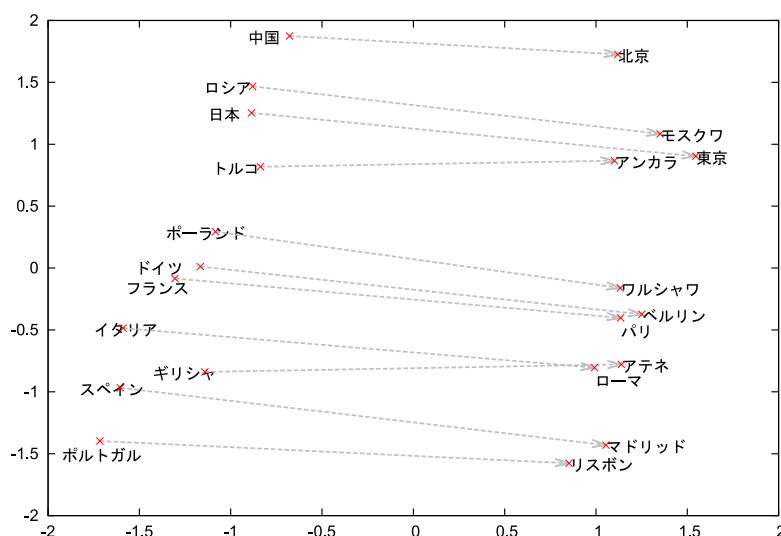
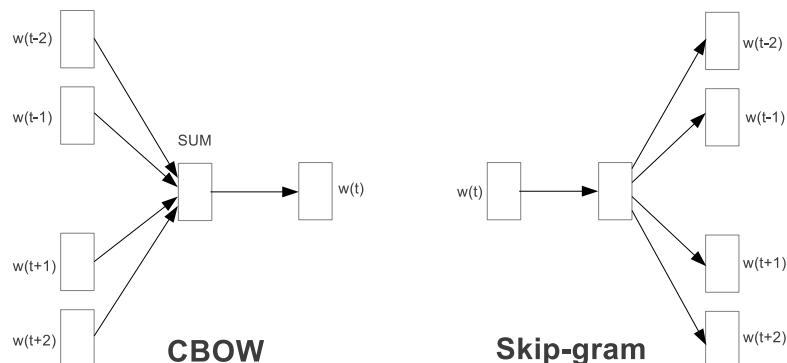
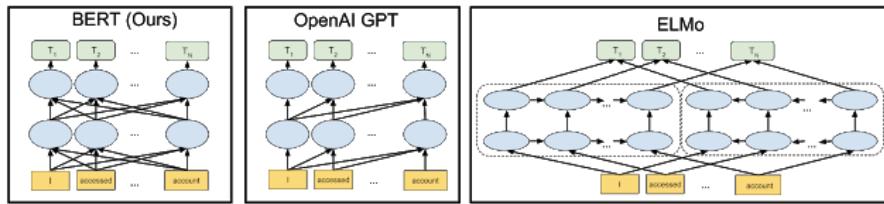


図6: 左図: 3単語対の性差を表す関係。右図: 単数形と複数形の関係。各単語は高次元空間に埋め込まれている
INPUT PROJECTION OUTPUT INPUT PROJECTION OUTPUT

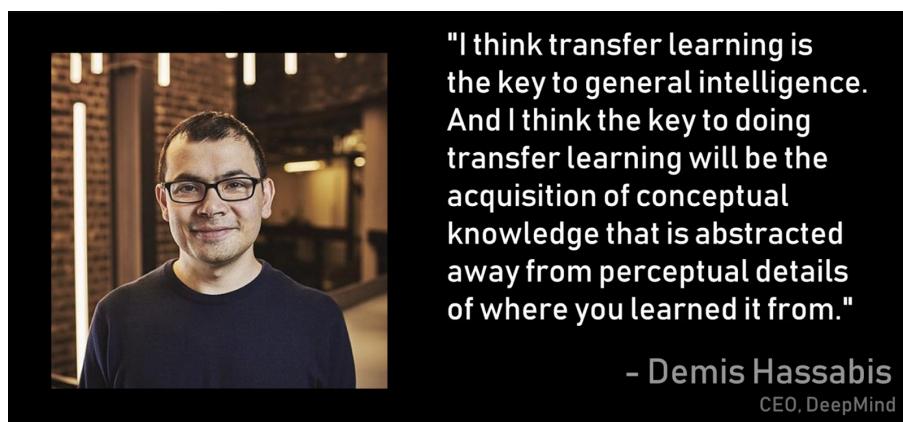


- word2vec の実習

センテンス埋め込みモデル



- 2018年は自然言語処理にとってのイメージネットブレークの時であったと言われたりします。すなわち人間超えしました性能で話題となりました。
- また最近炎上した GPT-2 に関する [open AI のブログ](#) も参照してください
- BERT 論文
- BERT の colaboratory デモ
- これらのモデルは一つのモデルで複数の課題に [転移学習](#) 可能であることを示しました。



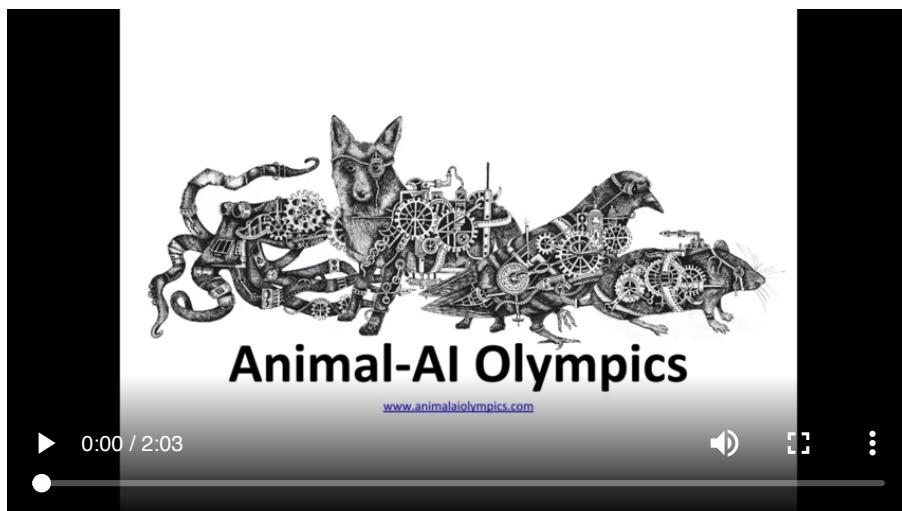
<https://towardsdatascience.com/transfer-learning-946518f95666>

強化学習 RL ロボティクス基礎

```
cd ~/study/2019tensorflow_models.git/research/a3c_blogpost  
# python a3c_cartpole.py --train  
python a3c_cartpole.py --algorithm=random --max-eps=4000
```

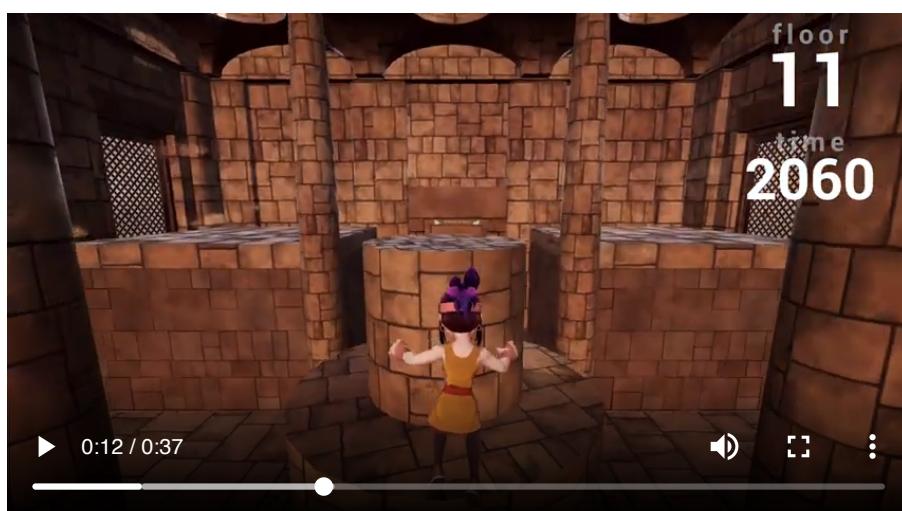
The Animal-AI Olympics

<http://animalaiolympics.com/>



Unity Obstacle Tower Challenge

<https://www.aicrowd.com/challenges/unity-obstacle-tower-challenge>



強化学習という言葉は古い言葉ですが機械学習の文脈では、環境とその環境におかれた動作主（エージェントと言ったり、ロボットシステムだったりします）が、環境と相互作用しながらより良い行動を形成するためのモデルです。動作主は、環境から受け取った現在の状態を分析して、次にとるべき行動を選択します。このとき将来に渡って報酬が最大となるような行動を学習する手法の一つです。

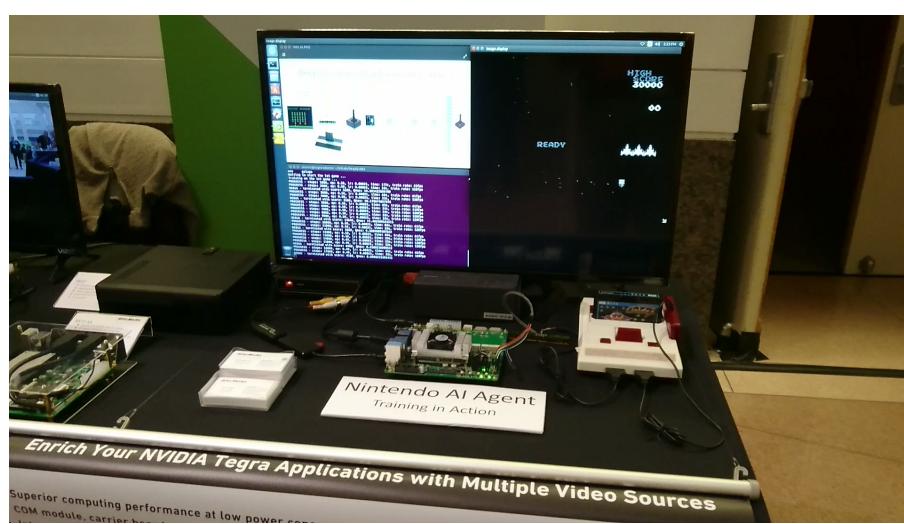
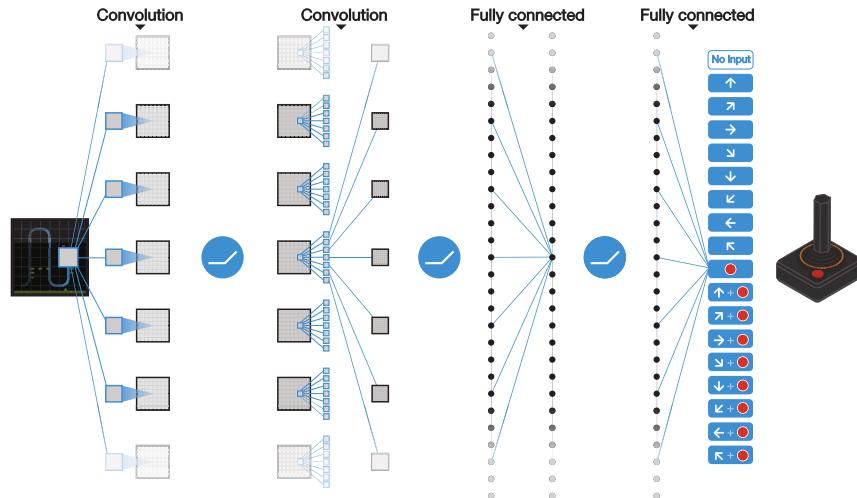
2015年には、Google傘下のDeepMindというスタートアップチームが開発した囲碁プログラムAlphaGoがプロ棋士のイ・セドル氏に勝利し話題になりました。AlphaGoは強化学習を基本技術の一つとして用いています。

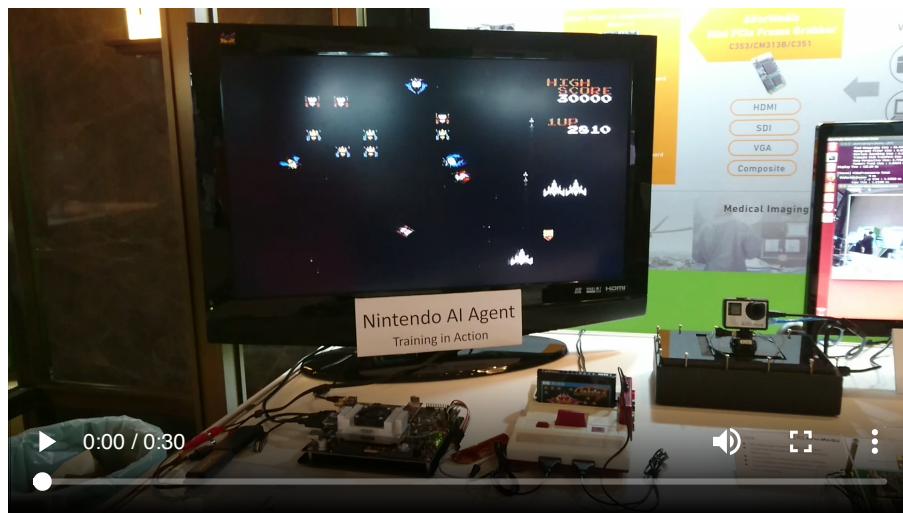
- エージェントと環境, マルコフ決定過程 MDP, POMDP, 効用関数, ベルマン方程式, 探索と利用のジレンマ, SARSA:
 - 値, 方策, Q学習, モデルベース対モデルフリー:
 - 深層Q学習:
 - ゲームAIへ(AlphaGo, AlphaGoZero, OpenAI five):
 - セルフプレイ:
 - 最近の発展A3C, Rainbow, RDT, World model:
-

複雑な状況をどう理解して解決するのか?

- 強化学習というニューラルネットワークモデルがあるわけではない
- 動的で複雑な環境に対処 → **強化学習 + DL** → 一般人工知能への礎
- DQN ATARIのビデオゲーム, <https://www.nature.com/articles/nature14236>
- AlphaGo囲碁, <https://www.nature.com/articles/nature16961>
- AlphaGoZero囲碁, <https://www.nature.com/articles/nature24270>

Deep Q Network





- **Q学習** Q learning に DNN を採用
- CNN が LeNet, @1998 LeCun そうであったように、強化学習 RL も昔からの技術 @Sutton_and_Barto1998
- ではなぜ、今になって囲碁や自動運転に応用できるようになったのか？
- ⇒ コンピュータの能力、データ規模、アルゴリズムの改良、エコシステム(ArXiv, Linux, Git, ROS, AMT, TensorFlow)

強化学習

- 強化学習 ⇒ 意思決定
- エージェント agent が 行動(行為) action をする
- 行動によって 状態 が変化する
- 環境 から与えられる 報酬 によって 目標 が決定
- 深層学習: ⇒ 表現、表象
- 教師信号として目標が与えられる
- 目標を達成するために外部状況の 表現 を獲得

強化学習 + 深層学習 = 人工知能

- 強化学習 ⇒ 目標の設定
- 深層学習 ⇒ 内部表象の獲得機構を提供

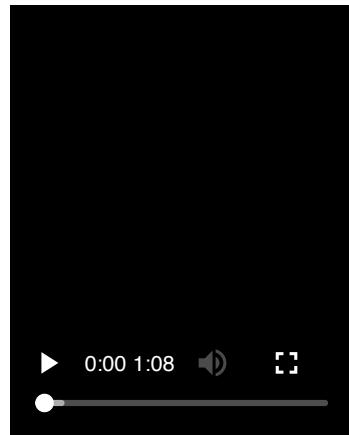
用語の整理

- 教師信号なし 報酬信号 reward signal
- 遅延フィードバック
- 価値 Value
- 行為 Action
- 状態 State
- TD 学習
- Sarsa
- Q 学習
- アクタークリティック
- 報酬 R_t : スカラ値
- 時刻 t でエージェントのとった行動を評価する指標
- エージェントは 累積報酬 cumulative reward の最大化する
- 報酬仮説: 目標は 累積期待報酬の最大化として記述可能

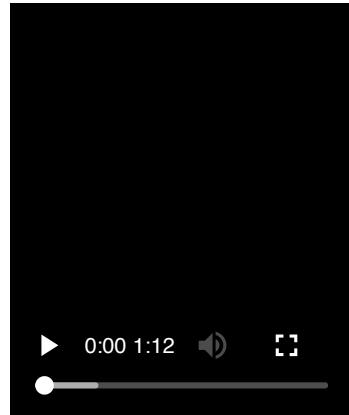
デモ

- ブロック崩し: <https://www.youtube.com/watch?v=V1eYniJ0Rnk>

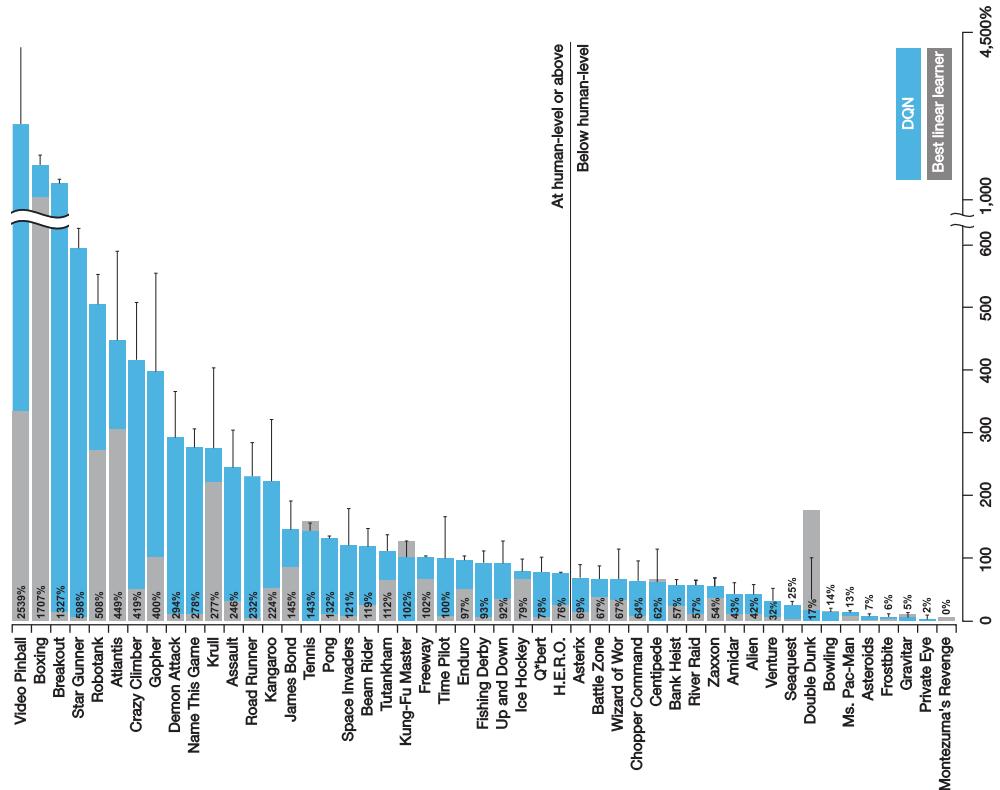
- ・スペースインベーダー: <https://www.youtube.com/watch?v=W2CAghUi0fY>
- ・OpenMind selfplay: <https://www.youtube.com/watch?v=OBcjhp4KSgQ>
- ・DQN の動画 スペースインベーダー



- ・DQN の動画 ブロック崩し



DQN 結果



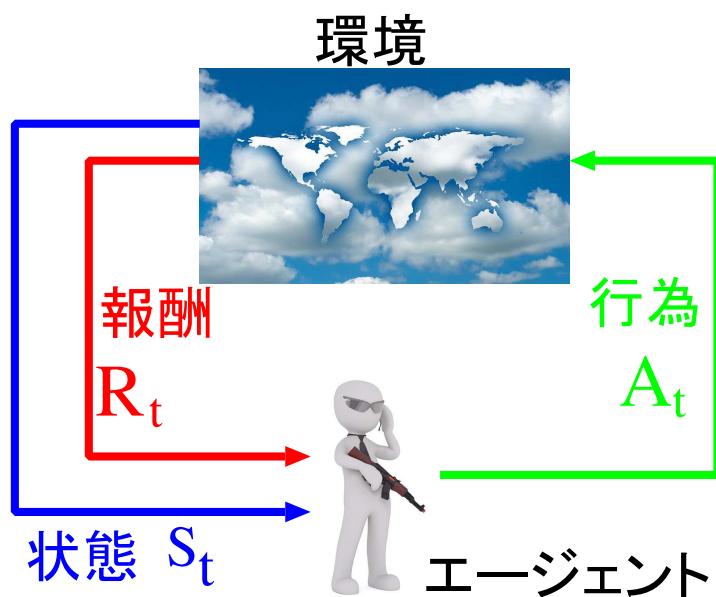
なぜ DQN には難しいのか？



人間にはできて強化学習には難しいこと

- Montenzuma's Revenge の動画 <https://www.youtube.com/watch?v=Klxxg9JM5tY>
- Private Eyes の動画 <https://www.youtube.com/watch?v=OfyS-Wj1M78>

エージェントと環境



- エージェント: 学習と意思決定を行う主体
- 行動 A_t を行い
- 環境の観察 O_t を行う
- 環境からスカラ値の報酬 R_t を受け取る
- 環境: エージェント外部の全て
- エージェントから行為 A_t を受け取り
- エージェントに観察 O_{t+1} を与え
- エージェントへ報酬 R_{t+1} を与える

エージェントの要素

- 方策 Policy
- 値値関数 Value function
- モデル エージェントが持つ環境の表象

方策 policy

- 方策: エージェントの行為
- 決定論的方策: $a = \pi(S)$
- 確率論的方策: $\pi(a|s) = p(A_{t=a}|S_{t=s})$

価値関数

- 将来の報酬予測
- 状態評価(良/悪)
- 行為の選択

$$v_\pi(S) = \mathbb{E}_\pi \left\{ R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots | S_{t=s} \right\}$$

強化学習のモデル

- 値値ベース
- 方策: なし
- 値値関数: あり
- 方策ベース
- 方策: あり
- 値値関数: なし
- アクター=クリティック Actor Critic
- 方策: あり
- 値値関数: あり
- モデルフリー
- 方策, 値値関数: あり
- モデル: なし
- モデルベース
- 方策, 値値関数: あり
- モデル: あり

探索と利用のジレンマ Exploration and exploitation dilemma

- 過去の経験から、一番良いと思う行動ばかりをしていると、さらに良い選択肢を見つけ出すことができない **探索不足**
- 更に良い選択肢ばかり探していると過去の経験が活かせない **過去の経験の利用不足**

目標, 収益, 報酬

- エージェントの目標は累積報酬を最大化すること (報酬仮説)
- 報酬仮説** Reward Hypothesis
- 目標: 期待報酬の最大化
- 時刻 t における報酬 R_t : スカラ値
- 時刻 t におけるエージェント行為の評価

逐次的意思決定 Sequential Decision Making

- 目標 Goal: 総収益を最大化する行動を選択すること
- 行為, 行動 Actions は長期的結果
- 収益は遅延することもある
- 直近の報酬を選ぶよりも, 長期的な報酬を考えた方が良い場合がある

収益 Return

- 収益** return G_t : 割引付き収益

$$G_t = R_{t+1} + \gamma R_{t+2} + \dots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

- 割引率 The discount $\gamma \in \{0, 1\}$: 現時点から見た将来の報酬を計算するため
- 遅延報酬** delayed reward の評価
- 0 に近ければ **近視眼的** 評価
- 1 に近ければ **将来を見通した** 評価

価値関数 Value Function

- 状態価値関数 v と 行動価値関数 q
- 状態価値関数 state-value function:

$$v_{\pi}(s) = \mathbb{E}_{\pi} \{ G_t | S_t = s \}$$

$$v_{\pi}(s) = \mathbb{E}_{\pi} \{ R_{t+1} + \gamma v_{\pi}(S_{t+1} | S_t = s) \}$$

- 行動価値関数 action-value function:

$$q_{\pi}(s, a) = \mathbb{E}_{\pi} \{ G_t | S_t = s, A_t = a \}$$

$$q_{\pi}(s, a) = \mathbb{E}_{\pi} \{ R_{t+1} + \gamma q_{\pi}(S_{t+1}, A_{t+1}) | S_t = s, A_t = a \}$$

最適価値関数 Optimal Value Function

- 最適状態価値関数:

$$v_{*}(s) = \max_{\pi} v_{\pi}(s)$$

- 最適行動価値関数:

$$q_{*}(s, a) = \max_{\pi} q_{\pi}(s, a)$$

- ベルマン方程式 一般に非線形になるので難しい

最適価値関数 Optimal Value Functions

- 最大の価値を与える関数

$$Q^*(s, a) = \max_{\pi} Q^\pi(s, a) = Q^{\pi^*}(s, a)$$

- 最適価値関数 Q^* が得られれば最適方策 π^* を求めることができる

$$\pi^*(s) = \operatorname{argmax}_a Q^*(s, a)$$

- 全ての意思決定における最適価値:

$$\begin{aligned} Q^*(s, a) &= r_{t+1} + \gamma \max_{a_{t+1}} r_{t+2} + \gamma^2 \max_{a_{t+2}} r_{t+3} + \dots \\ &= r_{t+1} + \gamma \max_{a_{t+1}} Q^*(s_{t+1}, a_{t+1}) \end{aligned}$$

- ベルマン方程式 Bellman equation:

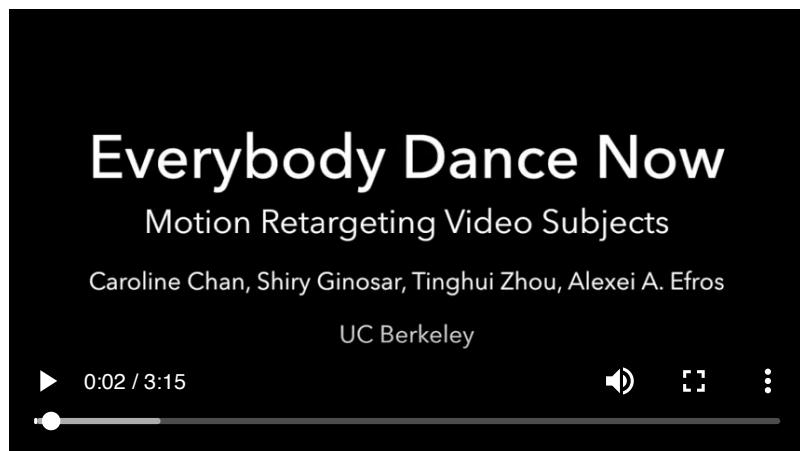
$$Q^*(s, a) = \mathbb{E}_{s'} \left\{ r + \gamma \max_{a'} Q^*(s', a') | s, a \right\}.$$

教科書, 参考文献

- An Introduction to Reinforcement Learning, Sutton and Barto, 1998, MIT Press, 1998
<http://incompleteideas.net/book/the-book.html>, Sutton and Barto (1998) (2018年第2版出版) 翻訳 強化学習 <https://www.amazon.co.jp/dp/4627826613>
- Algorithms for Reinforcement Learning, Szepesvari, Morgan and Claypool, 2010
<https://sites.ualberta.ca/~szepesva/papers/RLAlgsInMDPs.pdf>
- デービッド・シルバーの講義 <http://www.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html>
- ジョン・シュルマンのビデオ講義 <https://www.youtube.com/watch?v=oPGVsoBonLM>
- 「これからの強化学習」 <https://www.amazon.co.jp/dp/4627880316/>
- Algorithms for Reinforcement Learning, Szepesvari, Morgan and Claypool, 2010
<https://sites.ualberta.ca/~szepesva/papers/RLAlgsInMDPs.pdf>

GAN と VAE

- 実習:信長の夢



- Yann LeCun (現 FAIR 所長) said in [CMU RI Seminar: Yann LeCun: The Next Frontier in AI: Unsupervised Learning, November 18, 2016](#)

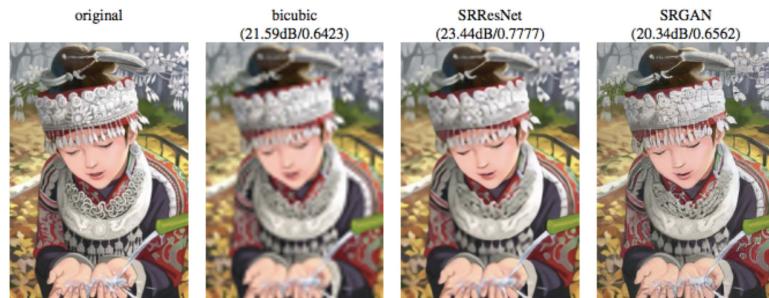
42分20秒頃から 敵対学習は本当にクールなアイデアだ。機械学習の分野で20年来で一番クールだ 'An adversarial training is a really really really cool idea. It's like the coolest idea in machine learning last twenty years.'



敵対訓練

- ざっくりいうと
 - 2人ゼロサムゲーム 古典的なミニマックス戦略
 - プレーヤーは双方ともニューラルネットワーク
- GAN は詳細高密度な画像を生成可能
- GAN はデータの密度関数を学習
- 少数の教師ありデータから、インタラクティブに種々の事例を生成可能 → さまざまな発展の可能性、プライバシー、製品開発、検査、

Single Image Super-Resolution



(Ledig et al 2016)

(Goodfellow 2016)

from Goodfellow(2016)NIPS tutorial

Image to Image Translation



(Isola et al 2016)

(Goodfellow 2016)

from Goodfellow(2016)NIPS tutorial

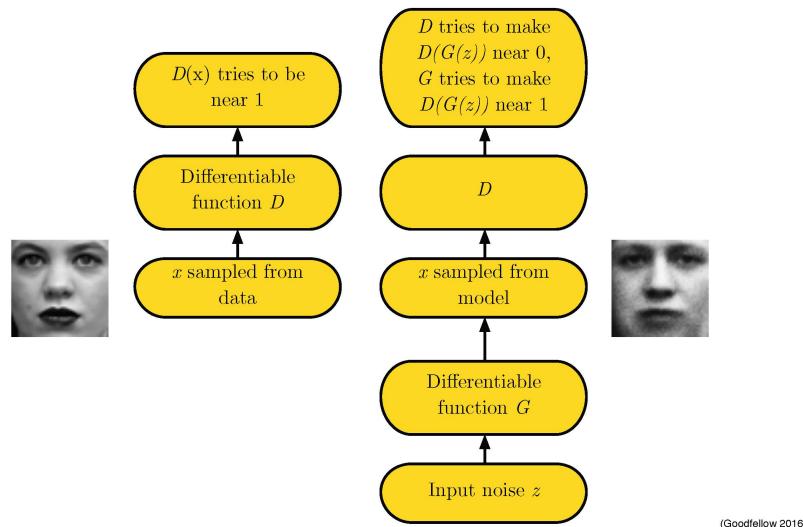
GANs

- Use a latent code
- Asymptotically consistent (unlike variational methods)
- No Markov chains needed
- Often regarded as producing the best samples
 - No good way to quantify this

(Goodfellow 2016)

from Goodfellow(2016)NIPS tutorial

Adversarial Nets Framework

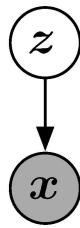


(Goodfellow 2016)

from Goodfellow(2016)NIPS tutorial

Generator Network

$$\mathbf{x} = G(\mathbf{z}; \boldsymbol{\theta}^{(G)})$$



- Must be differentiable
- No invertibility requirement
- Trainable for any size of \mathbf{z}
- Some guarantees require \mathbf{z} to have higher dimension than \mathbf{x}
- **Can make \mathbf{x} conditionally Gaussian given \mathbf{z} but need not do so**

from

(Goodfellow 2016)
Goodfellow(2016)NIPS tutorial

Minimax Game

$$J^{(D)} = -\frac{1}{2} \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} \log D(\mathbf{x}) - \frac{1}{2} \mathbb{E}_{\mathbf{z}} \log (1 - D(G(\mathbf{z})))$$
$$J^{(G)} = -J^{(D)}$$

- Equilibrium is a saddle point of the discriminator loss
- Resembles Jensen-Shannon divergence
- Generator minimizes the log-probability of the discriminator being correct

(Goodfellow 2016)
from Goodfellow(2016)NIPS tutorial

Non-Saturating Game

$$J^{(D)} = -\frac{1}{2} \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} \log D(\mathbf{x}) - \frac{1}{2} \mathbb{E}_{\mathbf{z}} \log (1 - D(G(\mathbf{z})))$$
$$J^{(G)} = -\frac{1}{2} \mathbb{E}_{\mathbf{z}} \log D(G(\mathbf{z}))$$

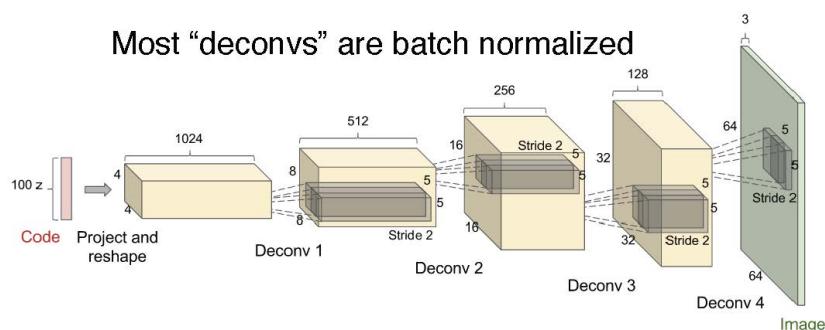
- Equilibrium no longer describable with a single loss
- Generator maximizes the log-probability of the discriminator being mistaken
- Heuristically motivated; generator can still learn even when discriminator successfully rejects all generator samples

(Goodfellow 2016)

from Goodfellow(2016)NIPS tutorial

DCGAN Architecture

Most “deconvs” are batch normalized

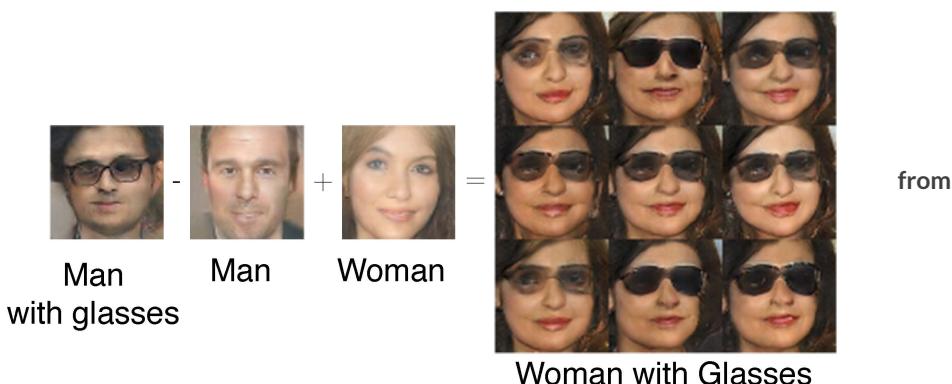


(Radford et al 2015)

(Goodfellow 2016)

from Goodfellow(2016)NIPS tutorial

Vector Space Arithmetic

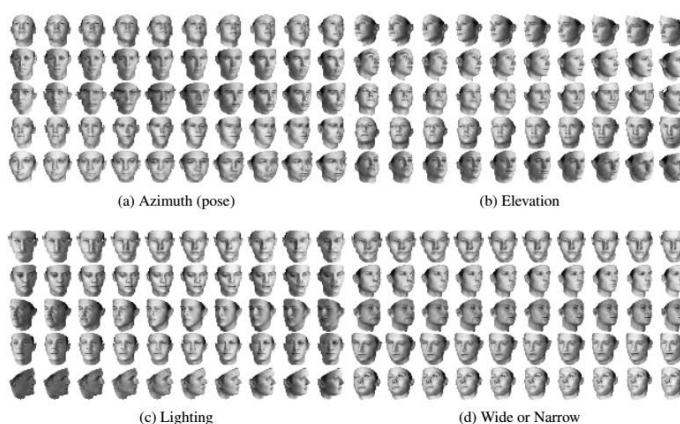


(Radford et al, 2015)

(Goodfellow 2016)

Goodfellow(2016)NIPS tutorial

Learning interpretable latent codes / controlling the generation process



InfoGAN (Chen et al 2016)

(Goodfellow 2016)

from Goodfellow(2016)NIPS tutorial

- [Collection of Interactive Machine Learning Examples](#)

Machine Learning Examples: Seedbank

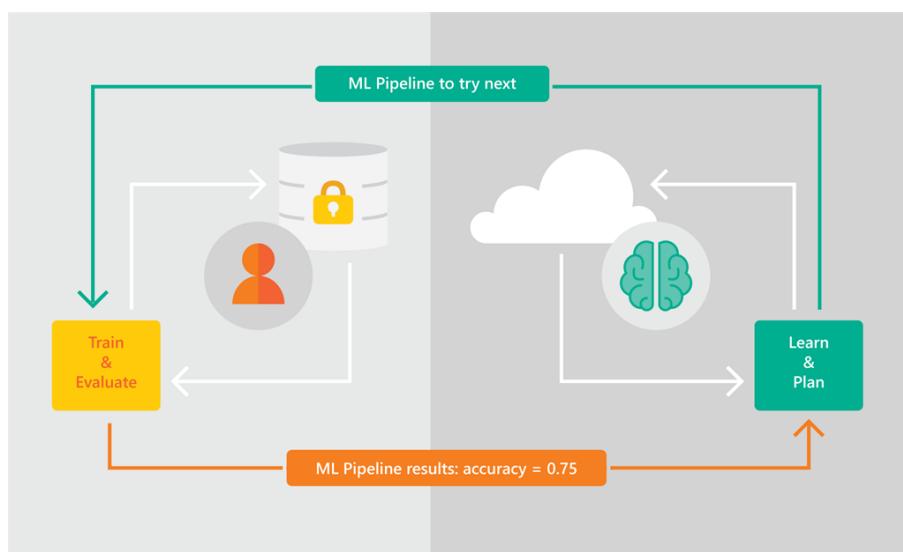
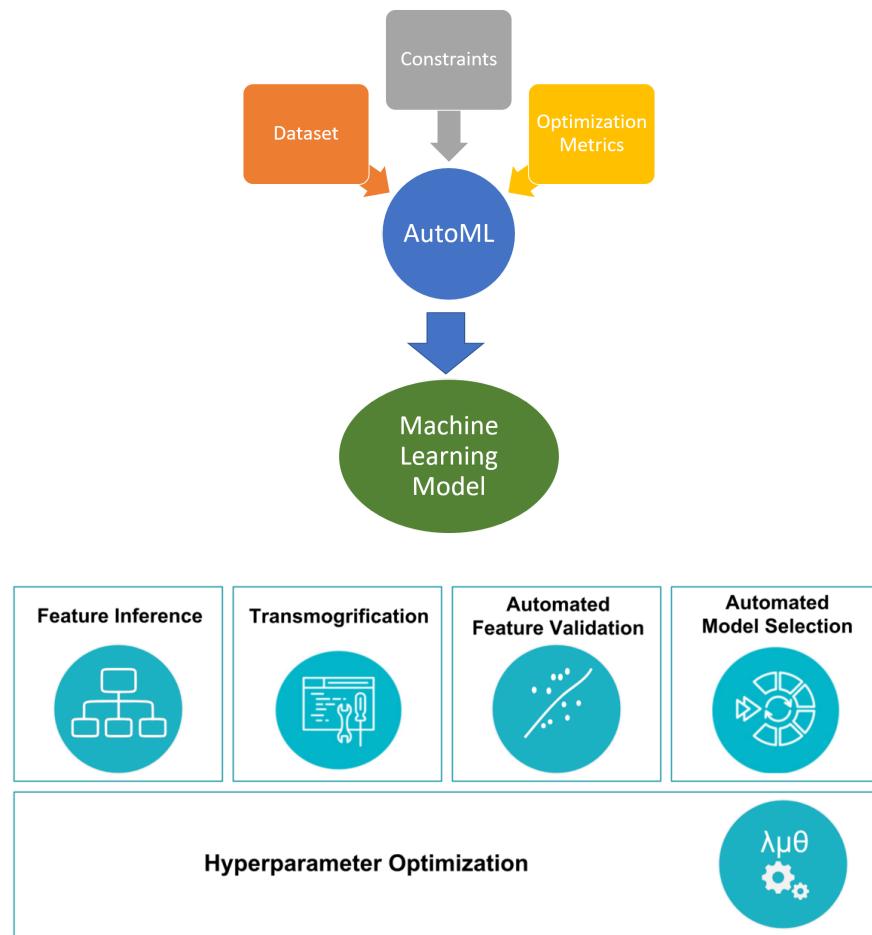
To see end-to-end examples of the interactive machine learning analyses that Colaboratory makes possible, check out the [Seedbank](#) project.

A few featured examples:

- [Neural Style Transfer](#): Use deep learning to transfer style between images.
- [EZ NSynth](#): Synthesize audio with WaveNet auto-encoders.
- [Fashion MNIST with Keras and TPUs](#): Classify fashion-related images with deep learning.
- [DeepDream](#): Produce DeepDream images from your own photos.
- [Convolutional VAE](#): Create a generative model of handwritten digits.
- [Convolutional VAE](#)
- [deep dream](#)

- [deep dream on google drive](#)

autoML

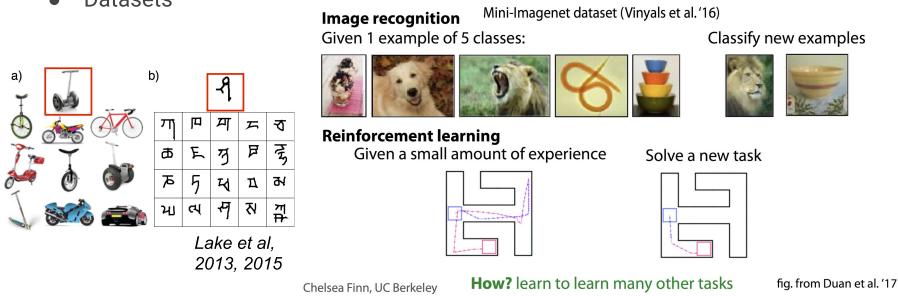


from medium [Using Machine Learning to Build Better Machine Learning](#)

- Salesforce.com TransmogrifAI: The Brain Behind Einstein
- Azure ML: Helping Developers Select the Right Machine Learning Model
- Waymo: Automated Model Selection for Self-Driving Vehicles
- [Using Evolutionary AutoML to Discover Neural Network Architectures](#)
- In “Large-Scale Evolution of Image Classifiers,” presented at ICML 2017, we set up an evolutionary process with simple building blocks and trivial initial conditions.

Learning to Learn

- What is Meta Learning / Learning to Learn?
 - Go beyond train/test from same distribution.
 - Task between train/test changes, so model has to “learn to learn”
- Datasets



from Read et. al (2017) NIPS tutorial

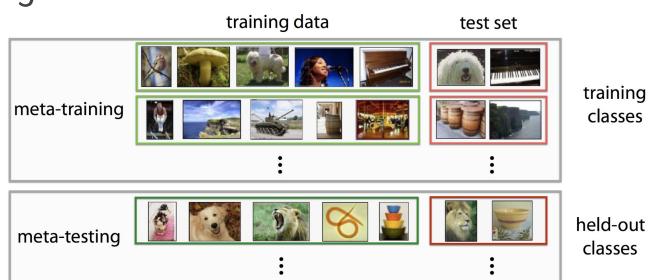
Learning to Learn



$$\theta = \arg \max_{\theta} \left[E_B \left[\sum_{(x,y) \in B} \log P_{\theta}(y|x) \right] \right].$$

from Read et. al (2017) NIPS tutorial

Learning to Learn



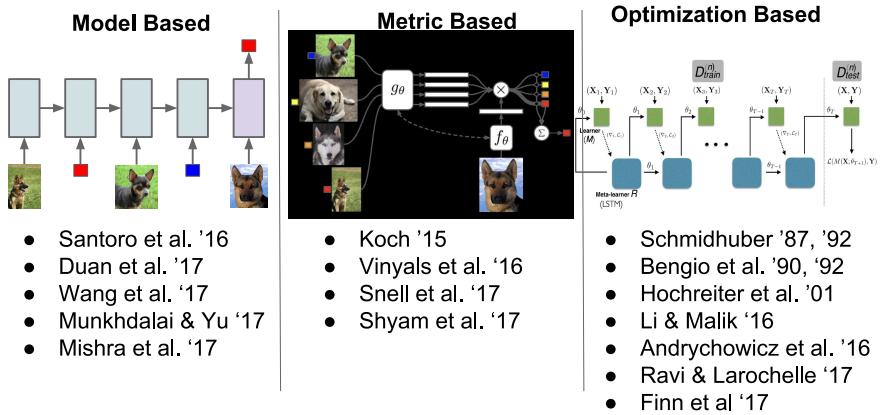
Chelsea Finn, Berkeley AI

diagram adapted from Ravi & Larochelle'17

$$\theta = \arg \max_{\theta} E_{L \sim T} \left[E_{S \sim L, B \sim L} \left[\sum_{(x,y) \in B} \log P_{\theta}(y|x, S) \right] \right].$$

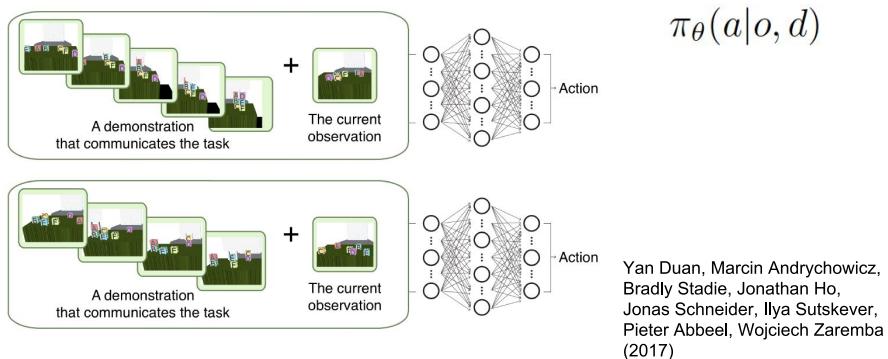
from Read et. al (2017) NIPS tutorial

Learning to Learn



from Read et. al (2017) NIPS tutorial

One-shot imitation learning: From a single demo d , learn to solve a new task as demonstrated.



from Read et. al (2017) NIPS tutorial