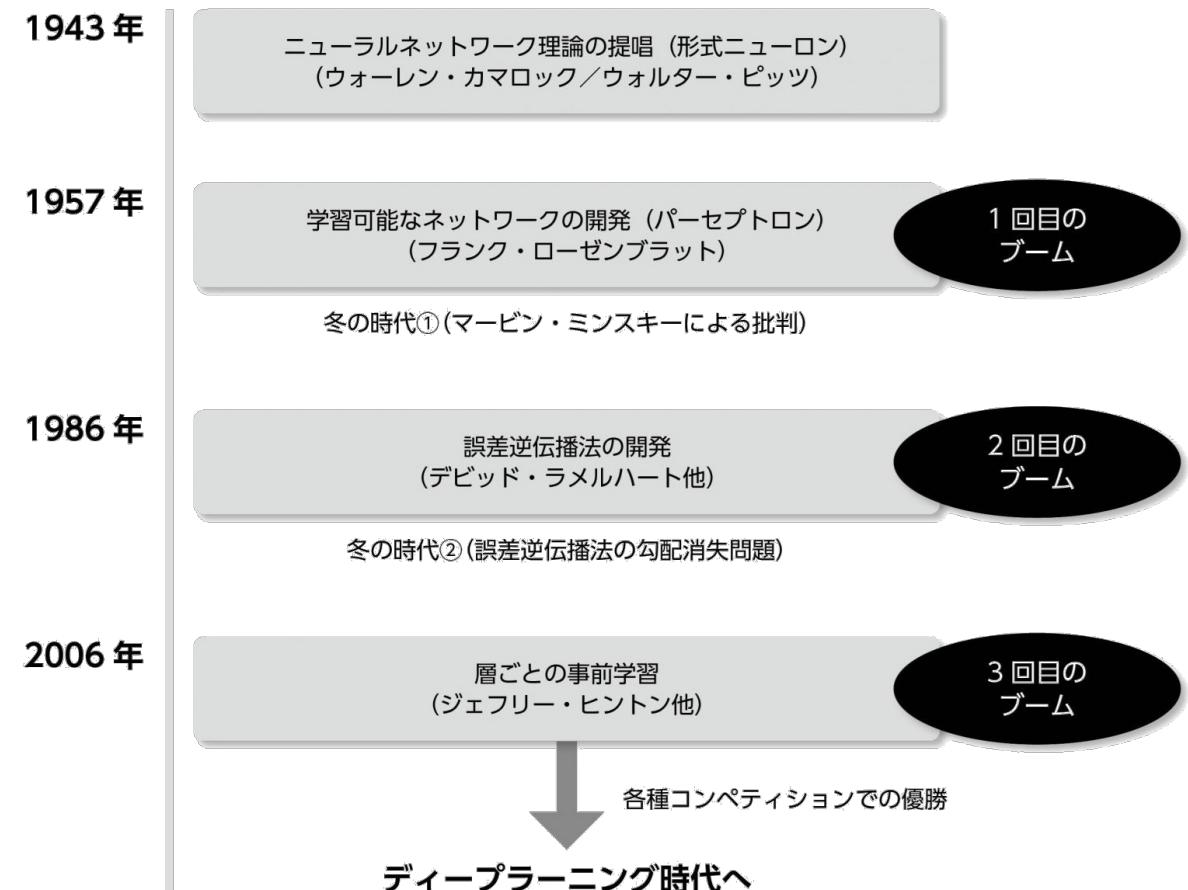
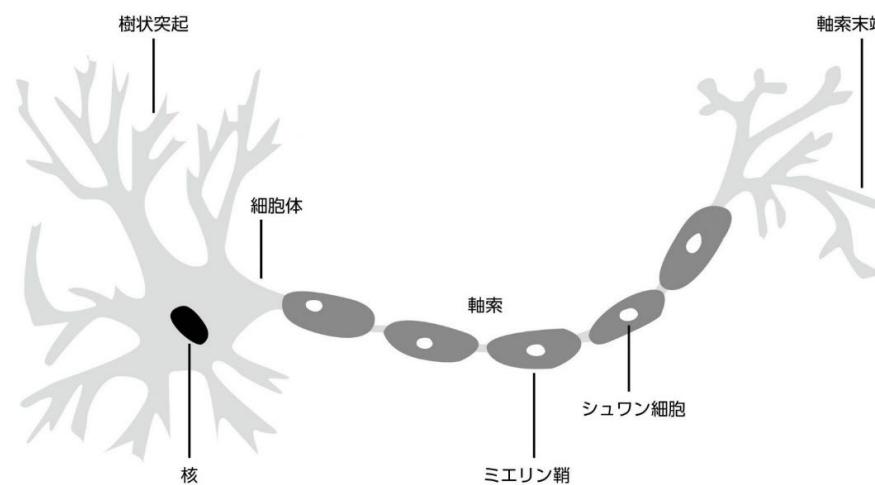




NNの歴史における第1次AIブーム

- 1943年のニューラルネットワーク理論の提唱に端を発し、1950年代ごろより計算機に探索や推論を行わせる研究が進む
- 脳や脳神経の生理学的研究が発端
 - 脳神経は大量にあり、それぞれが結合することでネットワークを構成
 - 様々な刺激を電気信号の強さで表現し相互に伝達している
 - 強い刺激が残ることで学習される

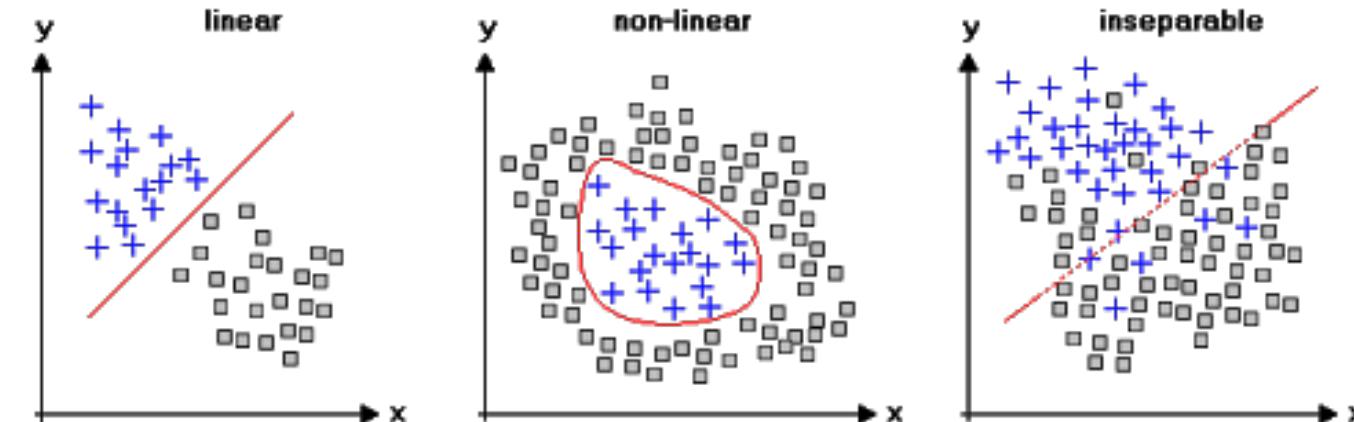




冬の時代①

- 人工知能研究の第1人者であるマービン・ミンスキーとケンブリッジ大で学んでいた数学者シーモア・パパートが、**パーセプトロンは線形分離可能な問題しか学習できないことを証明**

- n 次元空間上のふたつの集合を $n - 1$ 次元の超平面で分離できること
- 1970年代以降AI研究は進展がないと批判



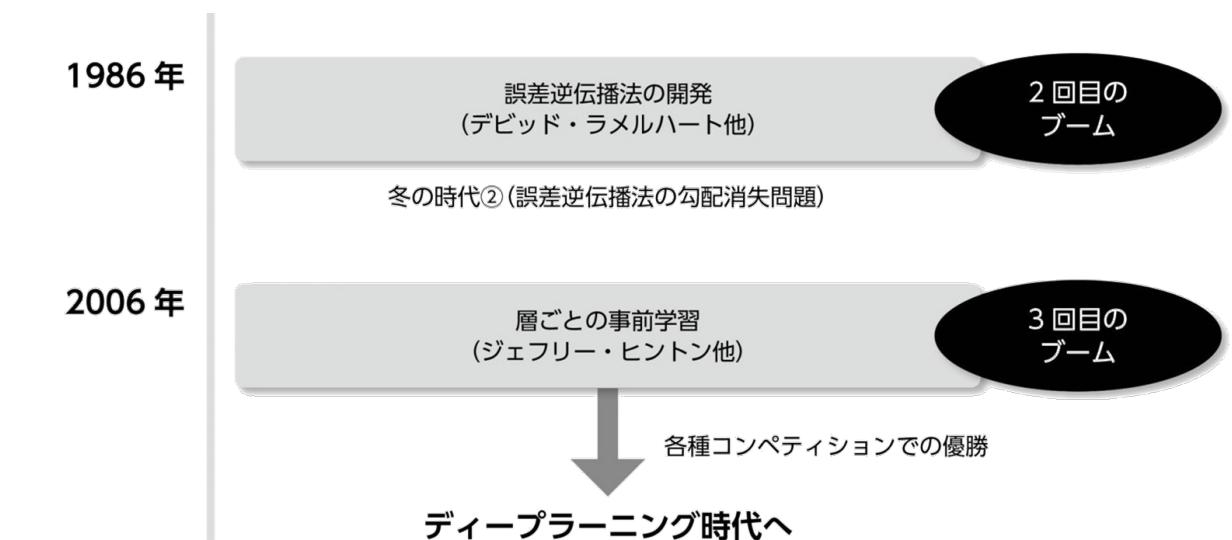
- さらに学生はロボットのはんだ付けばかりとこぼす
 - このころ花形であったのは電気系
 - 身体性と人工知能は両輪でロボティクスの発展は必要不可欠だという研究者もいる





第2次AIブーム

- 80年代には線形分離不可能問題を克服した多層パーセプトロンを発表
 - 入力層と出力層の2層構造であったのを隠れ層を追加した多層構造にする事で次元の境界を超えることができるため、線形分離不可能問題も学習できる
- 学習に相当する「逆誤差伝播法（バックプロパゲーション）」による多層での重み修正が可能に
 - Back Propagationについては別途
- 一部の探索問題では、極めて速く解を得ることができた
 - N-Queen問題など
 - 1990年頃は21 Queenが世界一
 - ルールベースではない学習における発展はみられなかった





冬の時代②

- ・沢山ニューロンを繋げると、誤差を伝搬させようとしても反映できる量がどんどん小さくなり、重みの変化を生み出すことができにくくかった
 - ・複雑にするとうまく学習が進まないため、簡単な応用しかできなかった
- ・行き詰まるNNをよそに、様々な研究が進展
 - ・k-means法、ランダムフォレスト法、サポートベクトルマシンなどの手法が発展し、優位性の主張が難しくなった
- ・記号接地問題（シンボル・グラウンディング問題）
 - ・我々は、水は透明・冷たい・べたべたするなどの「理解」や「概念」を持つ
 - ・コンピュータは「理解」することや「概念」の獲得が難しい
- ・これを真正面から取り組むのがオントロジー研究
 - ・分野によっては応用が進んでいる
 - ・しかしながら、人間の持つ概念を説明しきるに至っていない（あきらめムード？）





なぜ日本は遅れたか？

12

- 1990年代大学で学んでいるときは「エキスパートシステム」と呼ばれる推論エンジン研究が終盤に差し掛かっており、情報を集めて「どうするべきか？」「なにが原因か？」を探る研究の大反省が行われていた
 - 新世代コンピュータ技術開発機構ICOT : Institute for New Generation Computer Technology
 - この当時「通産省」の一大税金投入プロジェクトは失敗に終わった、時期尚早であった？
 - このトラウマから特に日本ではAI離れと懸念が浸透した（これが日本遅れの一因？）
 - 実はICOTで学んだ人、参加した超若手が、今のKコンピュータや富岳を作っている！
- このあたりは「Wikipedia 第五世代コンピュータ」が比較的詳しい
 - 西は渕先生に推論マシンPIMとそのオペレーション言語KL-Iについて講義を受けた
 - ICOTを欧米に伝えて大いに煽った米国人工知能学者ファイゲンバウムの談話
 - 第5世代は、一般市場向けの応用がなく、失敗に終わった。金をかけてパーティを開いたが、「客が誰も来なかっただようなもので、日本のメーカーはこのプロジェクトを受け入れなかった。技術面では本当に成功したのに、画期的な応用を創造しなかったからだ。」
- ICOTは負の遺産という側面は否定できないと考えるが、日本の人工知能研究における歴史を語るうえで避けて通れない





第3次AIブーム

- DL(深層学習)の発展により解ける問題のレベルが各段に上がった
 - 2012年画像認識コンペILSVRC優勝のAlexNet、AlphaGo、Watsonなど
 - 「データから特徴を抽出、経験則的に高精度で予測・分類を行う機能」が実現
 - 後で述べるが、DLにまつわるNN構成技術の発展だけではない（実は少ない？）
- ビックデータと呼ばれるデータ流通量の爆発的増加
 - インターネットの普及とIoTにより様々なデータが取得・流通できるようになった
- 無視してはいけないのが計算能力の向上
 - GPGPUの技術がDNNを加速、その加速の元になったのは1990年台のGPUの発展
 - GPUを加速したのはGUIの発展や、3D CADやゲーム需要の増加、SGIという企業
- あれ？日本はゲーム機産業ではトップを走っていたのでは？
 - 確かに初期の民生用ではそうであった
 - Family ComputerのPPUは8bitコンピュータ時代の傑作(BackGroundとSprite)
 - PlayStationでGTE(Geometric Transfer Engine)より3DG対応
 - バブル崩壊後のハイエンド需要時代において、フラッシュメモリ同様需要も見いだせなかった
 - その時期にSGIという企業がハイエンド系をOpenGLとアクセラレータで席捲した





機械学習からDLへ

- 機械学習

- 開発者があらかじめ全処理をプログラムせず、プログラム自身が解析し、法則性やルールを見つけて出す、つまりトレーニングにより特定のタスクを実行できる手法

- ディープラーニング

- 機械学習の発展形で、DNNを用いた手法のこと
 - 特に言語的な特徴表現が難しい場合に効果を發揮する
 - 要するに説明する必要がないということ
 - これを収束させる様々な技術が生み出された
 - 活性化関数ReLUの発明
 - 過学習回避手法の発明
 - Cross Validation(交差検証)
 - 正則化、L1正則化=Lasso回帰:異常データの重みを0へ、L2正則化=Ridge回帰:重みを0に近づける
 - Batch Normalization
 - AIC (赤池情報量基準) ・ BIC (ベイズ情報量基準) 、モデルの複雑さとデータとの適合度とのバランスを取る手法、オッカムのカミソリ(必要以上に仮定を増やすなということ、ケチの原理)は完全に無用化された
 - 様々な最適化、関連手法の発達
 - Dropout(最近はそれほど利用されない)
 - GAN (Generative Adversarial Network) 敵対的生成ネットワーク





そもそも第3次AIブームは終わったのか？

15

- 未だに発展途中な部分も多い
 - そもそも説明しなくてもよいようにしたのに…
 - 「DLは説明できないからダメだ」というのはなんだかね、そして逆変換を求めるのが極めて困難で求めることもできていないのに「説明可能」とするのもなんだかね、という状態
- またしても誤解が多かった
 - ディープラーニングが万能とする意見はかなり減ったがそれでも過剰評価は多い
 - 成長限界が来ているという意見もある
 - Filip氏によるDLはスケールしないという指摘は著名かつ重要
 - 「計算コストが100倍1000倍と必要になるが、精度はそれに見合った上昇を遂げない」
 - 様々な意見
 - AIでできることとできないことが明確化
 - 新たなイノベーションが生み出されておらず、プラトーの状態
 - ディープフェイクなど、AIについて「不吉な技術」だと考える人が増えている
 - 残念ながらAI投資の終焉が人気の終焉、再び冬の時代になる
 - だから学ばないのでない、学んで正しい道を選択できるようにすること





実際にAI・MLについて学ぼう

16

- AI・MLが特別な技術というわけではない
 - ほぼリソースと運が勝負の世界？使って、改良できるだけでも十分
- そもそも、本格的に学びたいならKaggle(www.kaggle.com)などをやるとよい
 - 授業は学生万人受けを狙わなければならず、「新たなデータに対して自ら問題を発見し解決する能力」はつかないし、高々20時間ぐらいで学習できるわけもない
 - 「The Home of Data Science & Machine Learning」とある通り世界の機械学習・データサイエンスに携わる約40万人が集まるコミュニティ
 - コンペもあり、アルバイト代わりに解くことできる
 - まずは、Kaggle Kernelで様々なデータやコードが手に入る(すべて英語です)
 - 研究室の研究そっちのけでKaggleするKagglerもいるし、そういう学生ほど初任給がとんでもない企業に入っていく、中退する学生もいるが、まあ、それもよいだろう
- 先人がNNやMLを扱う上で必要かつ十分な道具を準備してくれている
 - これらを使って学ぶことを目指します
 - 皆さんにとって必要な環境は「ブラウザ」です
 - とはいえてあまり過度に期待されても困ります





そもそも、DNNで何ができる・できそうか？

17



自動運転

勝手に障害物となる人や対向車などを認識して、安全に避けてるなどし、事故を回避する

では、なぜそれにDNNが必要か？

従来手法よりも精度よく対象物を認識する必要があり、それを達成した



スマートフォン

色々なアプリがDNNを用いて、顔やそのパーツを認識したり、写真から邪魔なものを消去できる

では、なぜそれにDNNが必要か？

特に何かパラメータを与えなくても、消去したい物体を指定するだけで消去できるようになる



医療

レントゲンや心電図などから、医者に変わって疾患を見つけてくれたり、処置を指南する

ではなぜそれにDNNが必要か？

やはり精度よく見つける必要があり、それを達成したことに加え、処置の指南といったより発見的な対応も可能となつた





やっぱりすごいじゃないですか？！

18

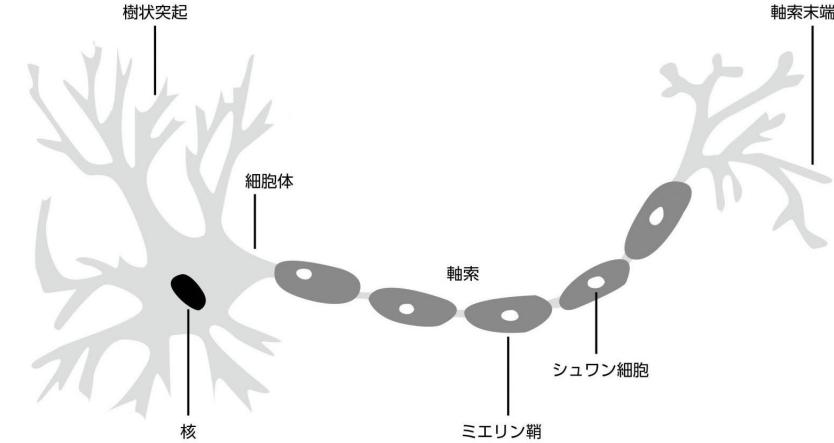
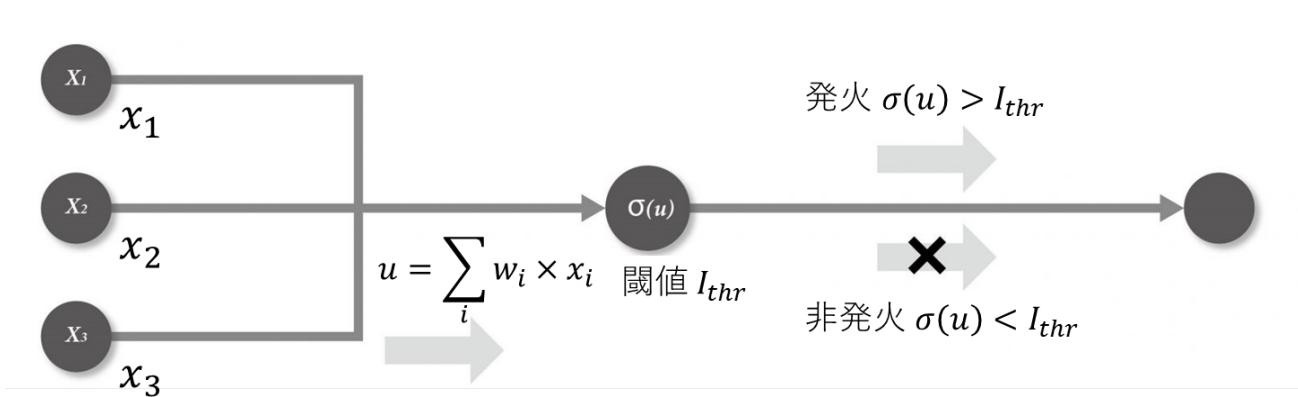
- 新聞のAIの発展で消えゆく職業といった記事はセンセーショナル
 - 教員は含まれていないので安心
 - ただし、本当にその通りになるかは誰もわからない
- AIの利用や、その波及効果に対する懸念は今に始まったことではない
 - 古くから様々な懸念があった
 - では、なぜ突然流行りだしたか？
- 敢えて言えば、突然ではない
 - 「技術革新」の積み上げがそこにあったから
 - ということで、簡単に歴史を紐解く



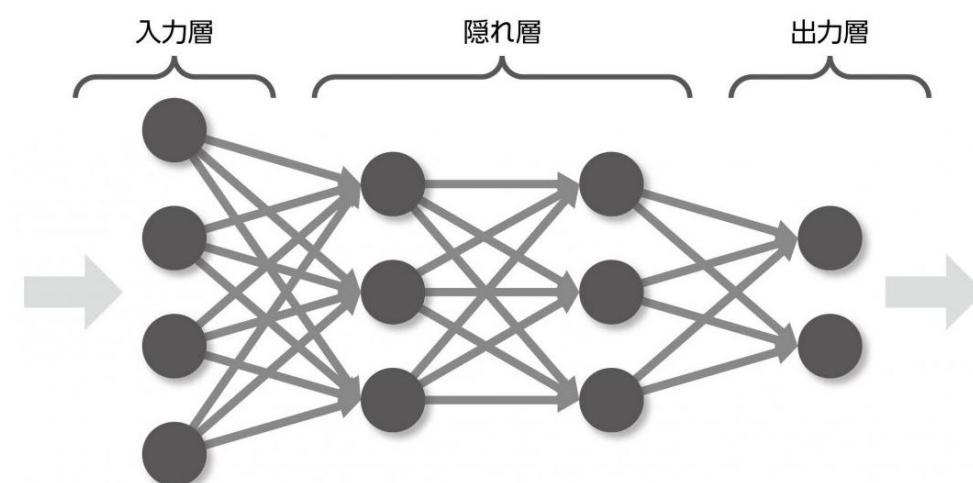
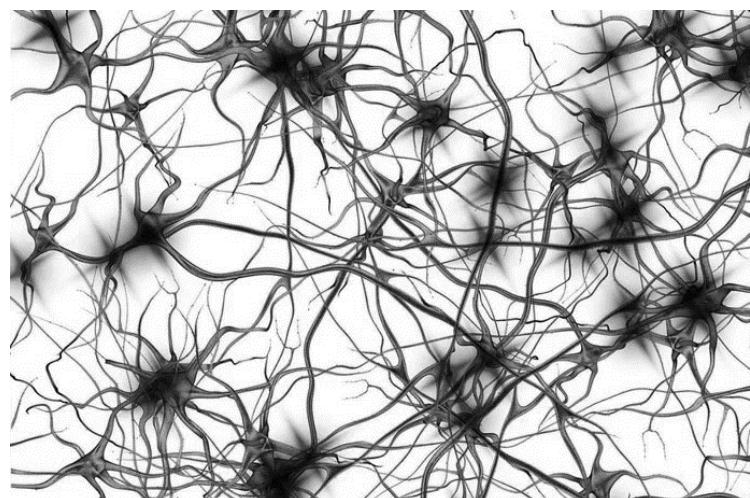


ニューロンの数理モデル

- 樹状突起が他のニューロンと接続、複雑に絡み合っている
 - そこから集めた刺激をもとに、軸索末端で次のニューロンに電気的刺激を伝える



- さらに脳のニューロンネットワークを模して多層で結合





利用する際には「学習」と「推論」を行なう

- 学習：学習用のデータ（教師データ）を用いてNNに含まれるニューロンの重み値を調整するステップ
- 推論：調整済みの重み値をもつネットワークを使って、回答を出すステップ

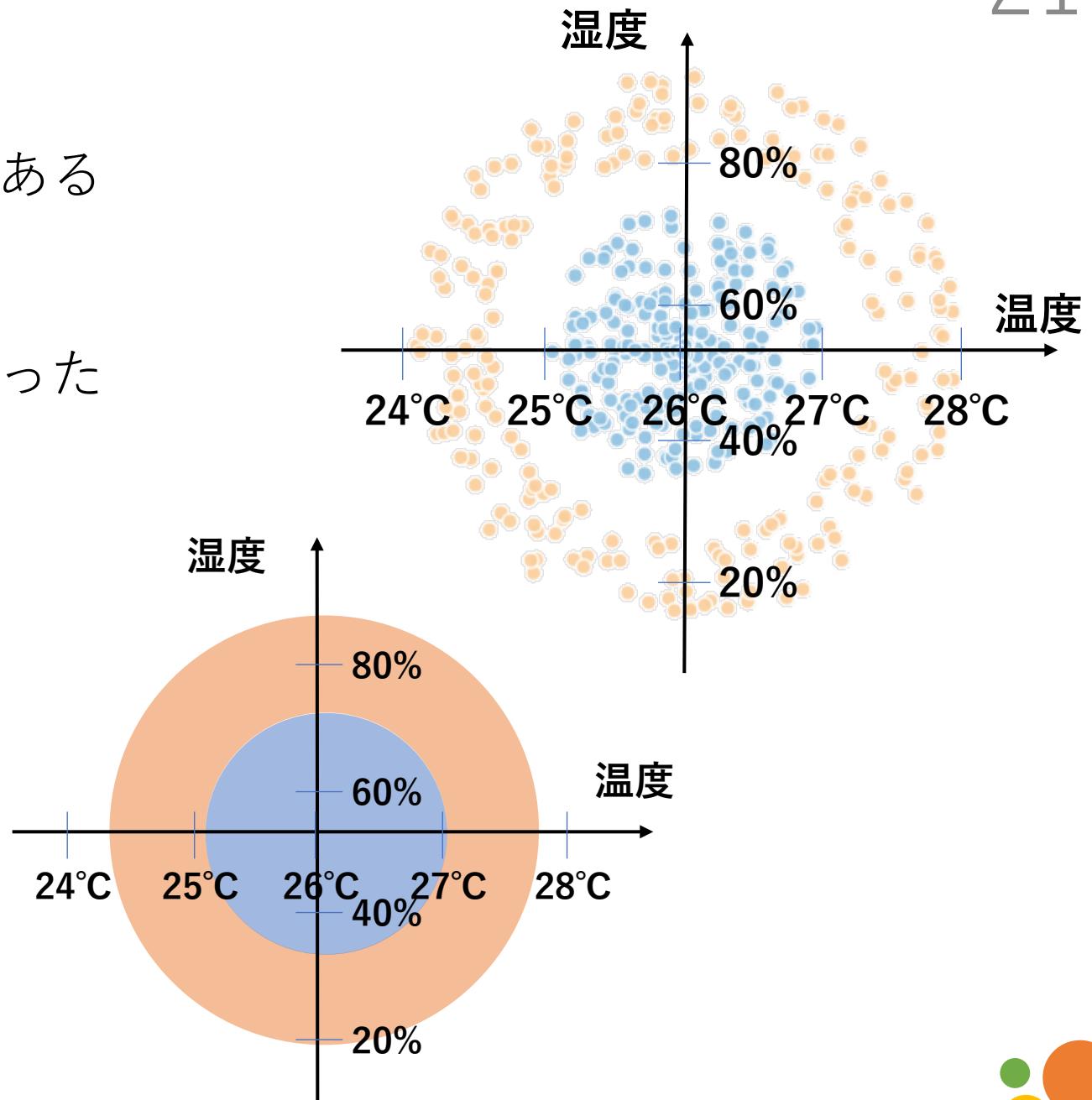
一般に、学習には大量の計算資源と時間が必要

- すなわち、計算機の能力向上がその発展には不可欠であった
- 結合情報や学習が進んだ重み値（纏めてモデルと呼ぶ）の再現は比較的容易にできるため、応用しやすい



具体的な例…

- エアコンを自動制御したい
- リモコンに快適ボタンと不快ボタンだけある
 - これを押させて情報を集める
 - その時の温度と湿度を監視する
- その申告値をグラフすると右のようになった
 - 快適と申告した時の温度と湿度に青点
 - 不快と申告した時の温度と湿度に赤点
- 人間ならば、
 - 右のように制御すればよいとすぐにわかる
 - 赤に入ったら制御！青に入ったら現状維持！
- 機械はどうする？
 - 人によって形は違う…





基本機能を備えた<https://bit.ly/3gZzHq5>で遊ぶ

22

<https://playground.tensorflow.org>

特徴量の選択

学習の実行

学習のやり直し

検証データの選択、決められたデータしか使えないのが残念だが、コードが公開されており変更できなくはない

全データを、学習用と検証用に分割するが、その割合

値にノイズを入れてデータに変化を与える

与えられたデータを何件ずつに分けて計算し重みを更新するかのバッチサイズを決める

Epoch: 000,000

Learning rate: 0.03

Activation: Tanh

Regularization: None

Regularization rate: 0

Problem type: Classification

DATA: Which dataset do you want to use? MNIST

FEATURES: Which properties do you want to feed in? X1, X2, X12, X22, X1X2, sin(X1), sin(X2)

+ - 2 HIDDEN LAYERS

+ - 4 neurons

+ - 2 neurons

OUTPUT: Test loss 0.520, Training loss 0.534

Ratio of training to test data: 50%

Noise: 0

Batch size: 10

REGENERATE

線の太さは重みの大小に比例

Colors shows data, neuron and weight values.

Show test data Discretize output

エポック = 試行回数
1エポック = データセットサイズ ÷ (ミニ)バッチサイズ回のイテレーション

学習率：パラメータを勾配法によって更新するが、その更新量を調整する係数、初期値はそのままでよいが0.01などが良く用いられる

活性化関数

隠れ層の数を変える

過学習を避けるための正則化についてL1ノルムかL2ノルムを選択

ノルムを足す割合、小さいと過学習大きいと学習不足

回帰(Regression)と分類(Classification)を選択、今回は分類つまり、赤と青の領域分割をやってみる

損失関数の値、目標と実際の差の指標で、学習データの値と、検証データの値を表示

途中経過や結果の表示



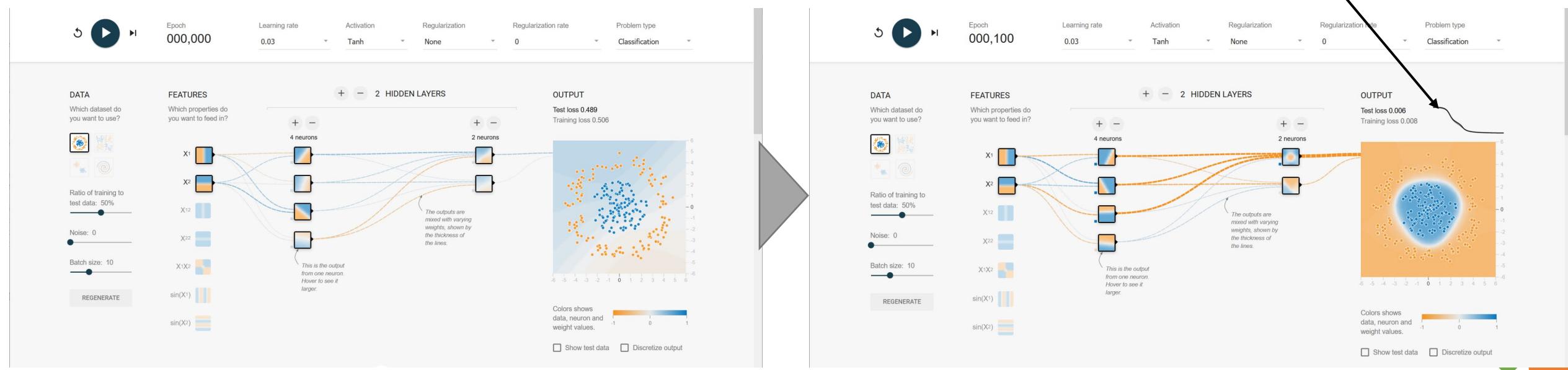


まずは最初の状態から

23

- ▶を押すと学習が進む
- 学習が進むと、損失関数の値が小さくなる
 - つまり、学習結果と教師データの差が少なくなり、その推移グラフも右肩下がりに推移する
- すると、先の例の「快適」と「不快・要制御」の領域が塗り分けられる
- 重要なのは「青や赤がない点も塗り分けていること」
- まずは、何ができるかを理解する
 - ハイパーパラメータを色々変えるのは、ちょっと後回し

- このグラフが結構大事
- 値が小さいほどよい！
 - 学習データと検証データを表示

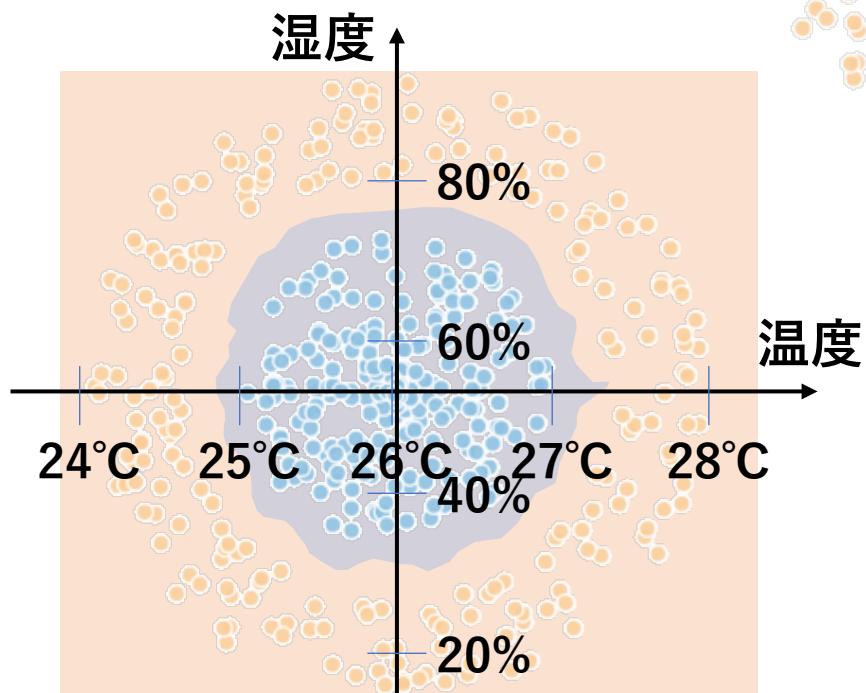
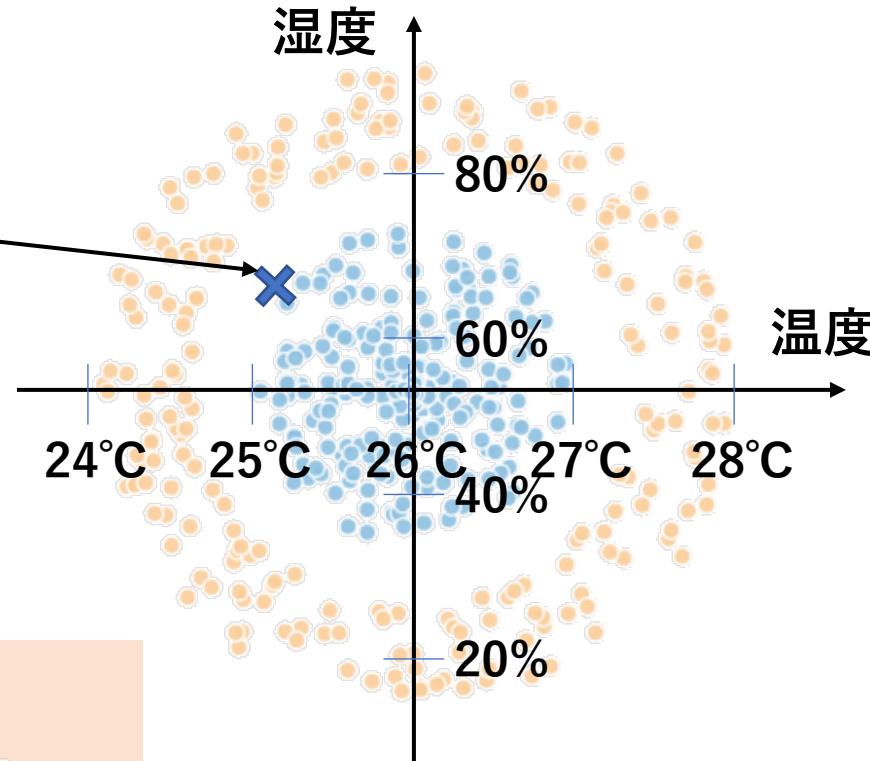




おいおい、これならもっと簡単にできるぞ

25

- その通り！
 - 例えば、この調べたい点がある
 - 周りの点との距離を2点間の距離の公式で調べて、「一番近いのは青点、赤点どちらか？」とか「近いトップ10位に青点が多いか、赤点が多いか？」とかを調べればよい
- このようなデータならばうまくいく
 - やってみたら、右の通り
 - 出来てはいるが…
- だが、現実はそうではない

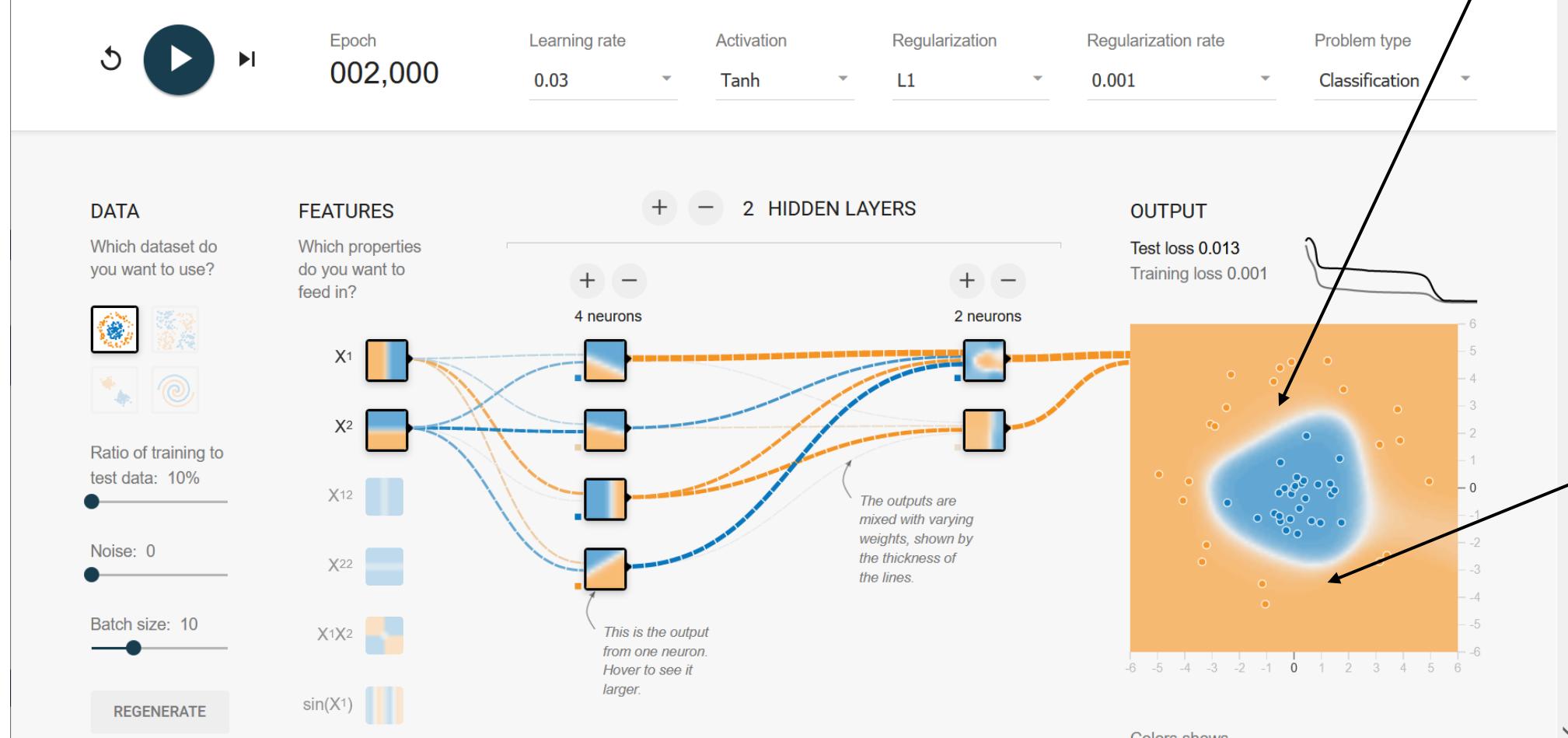


どうも…データが少ないのよね

26

- テストデータの数を10%と少なくしても、うまくいくように設定できる

人間が境界を決めたかのように滑らかに結ぶのが神

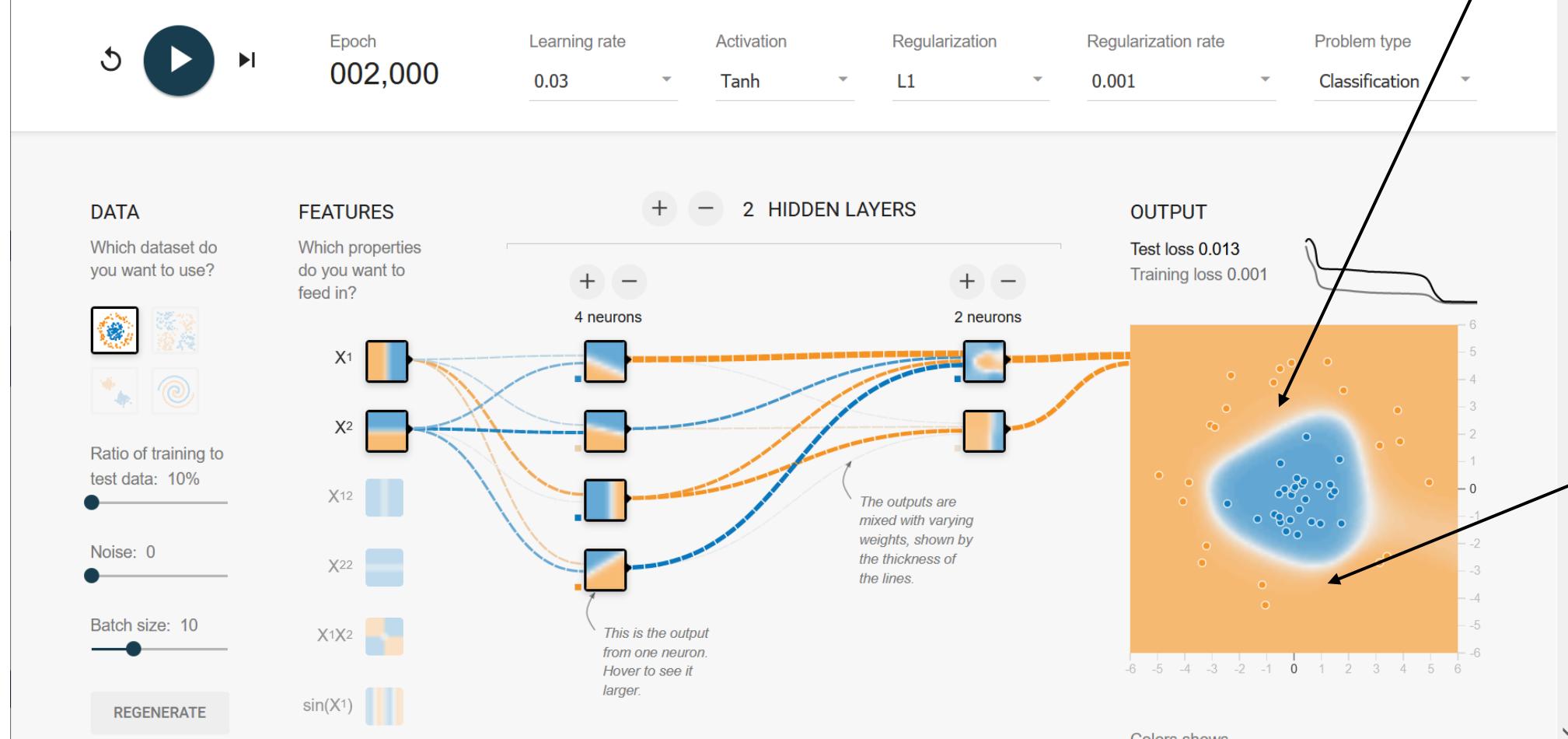


どうも…データが少ないのよね

27

- テストデータの数を10%と少なくしても、うまくいくように設定できる

人間が境界を決めたかのように滑らかに結ぶのが神

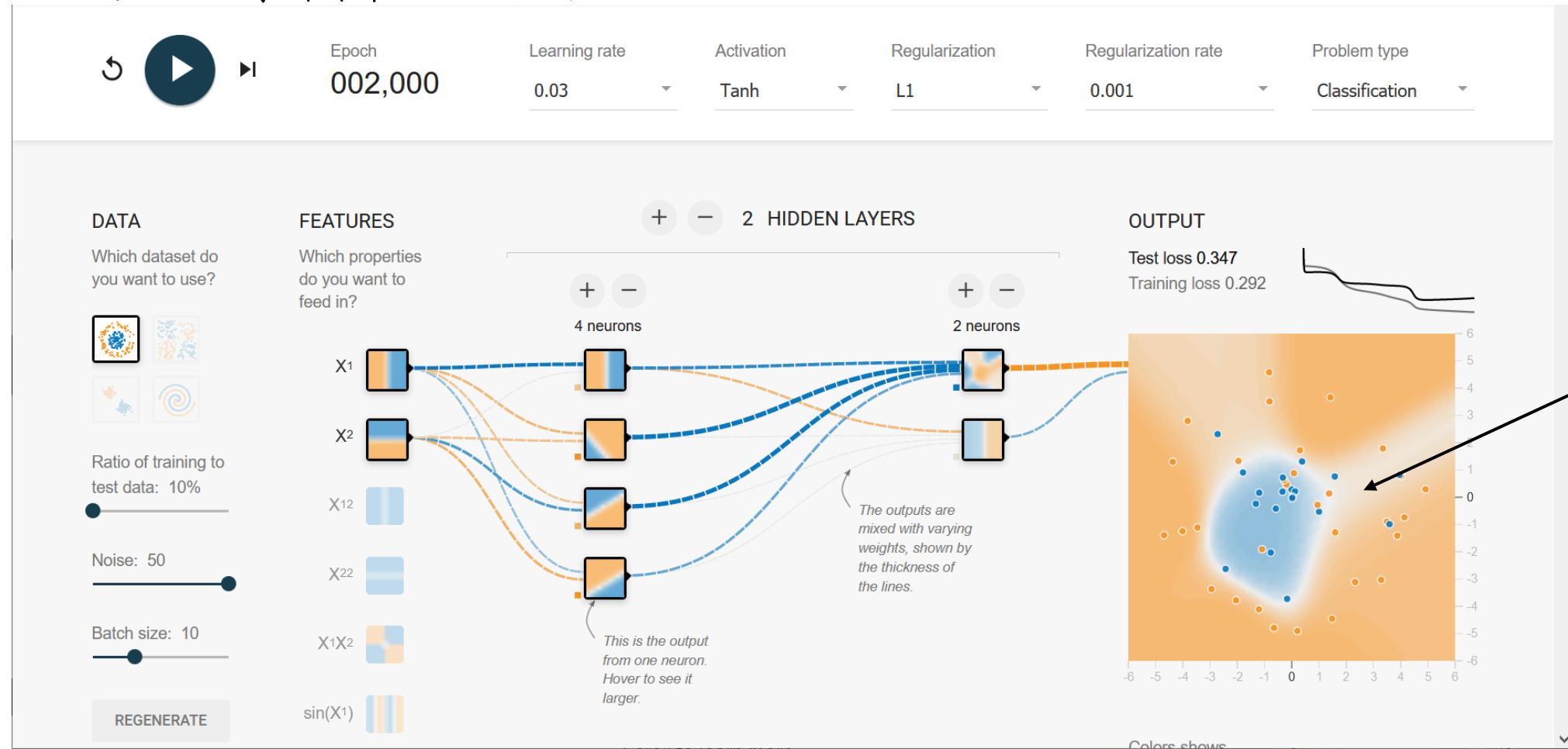




データも少ないし、人間って適當なのよね

28

- 人間って機械じゃないから…
 - 同じ温度湿度でも、家族喧嘩して「不快」と怒って押してみたり
 - AIだと？！いじめてやれ！って敢えて逆を押してみたり、寝ていて誤ってボタンを押してみたり
- それでも、出来てしまう



- ノイズがあってもきちんと領域を塗り分けている
- 人がやっても上手く境界を定めるのは難しいかも







つまり口バストである

30

- 非線形な場合でも対応する
 - 線形つまり、直線で表現できるのに、わざわざAIを使うのは大げさで、しばしば学会で笑いものになる
 - 芋を太陽に突っ込んで焼くな
 - 非線形な関係は、表現の幅が広すぎてグンと難しくなる
 - 世界は非線形なものばかり
- モデルがわからなくても、なんとかなる
 - 物理的にモデルがわかっている場合は使うな
- 少々ノイズがあっても、なんとかなる
 - 一言で言えば、「いい感じにやってくれる」

横文字ビジネス用語の早見表

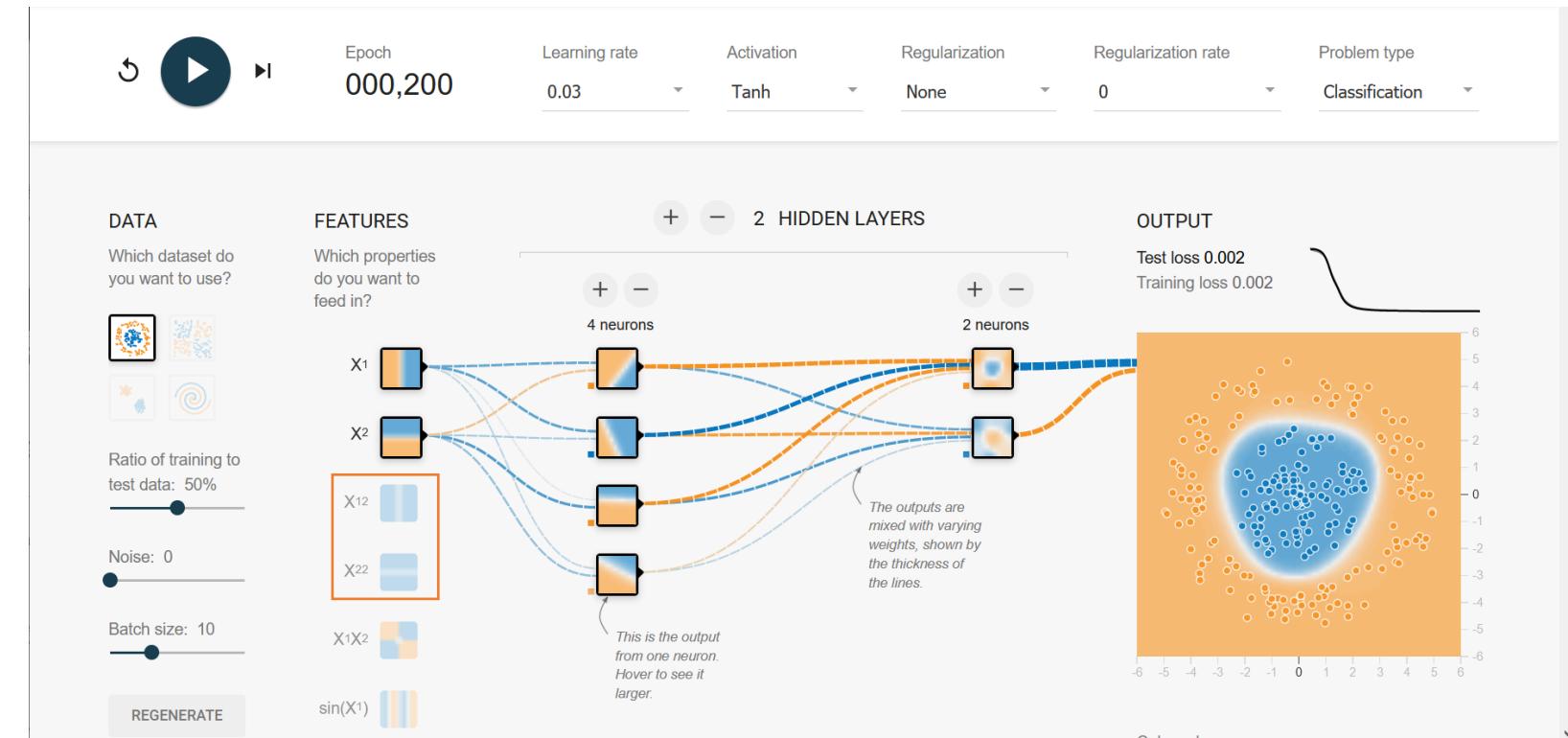
「イノベーションを起こす」 →	いい感じにやる
「PDCAを回す」 →	いい感じにやる
「ゼロベースで考える」 →	いい感じにやる
「ソリューションを提供する」 →	いい感じにやる
「フルコミットする」 →	いい感じにやる





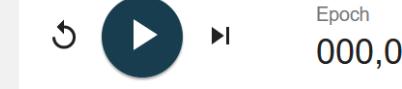
基礎的なことをまずは学ぶ

- 一番最初の状態から
- データをそのまま食べさせてもダメなことが多い
 - そのまま与えてもよいように工夫することもできるが、教師データが大量に必要になる
- 基本は、「特徴量を与えてあげる」
 - ここでは、温度 X_1 と湿度 X_2 で影響を受けるのだろうという「常識の値」が用いられている
 - では、特徴量を工夫すると、早く学習するのか？



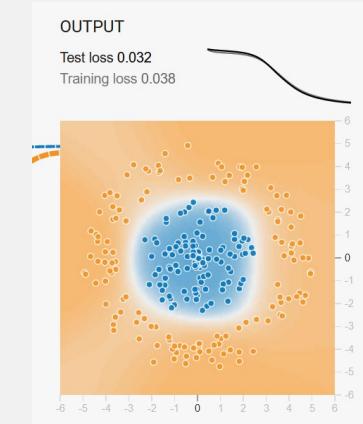
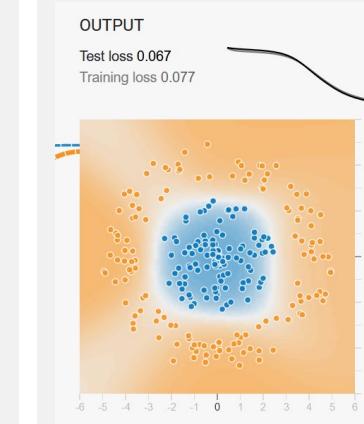
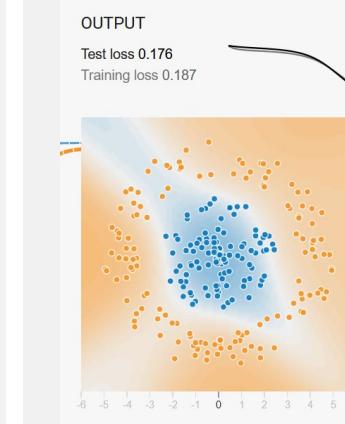
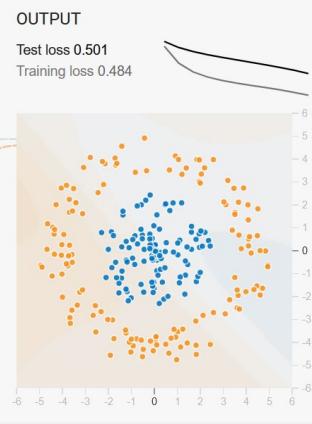
特徴量を工夫する

32



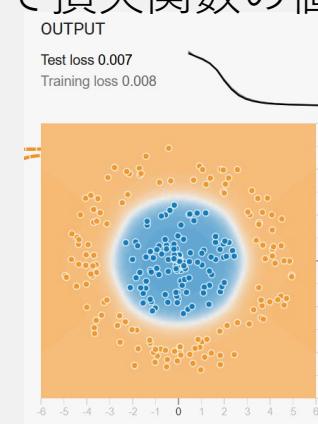
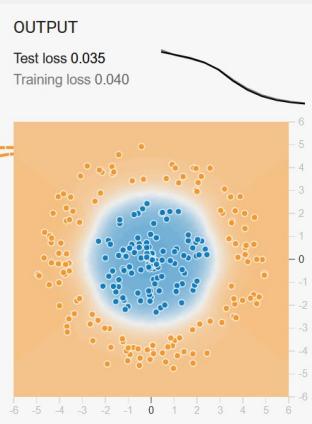
- そのまま X_1 と X_2 を与える (温度と湿度が影響するのよね、という情報)

ここで損失関数の値は 0.032



- 工夫して X_1^2 と X_2^2 を与えてみる (この辺りが普通の人は快適という情報)

ここで損失関数の値は 0.007



圧倒的に早く収束している = 学習が進んでいる！





なぜ χ^2 という関数を選んだのか？

33

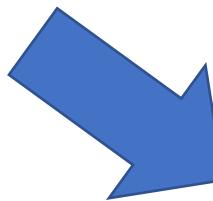
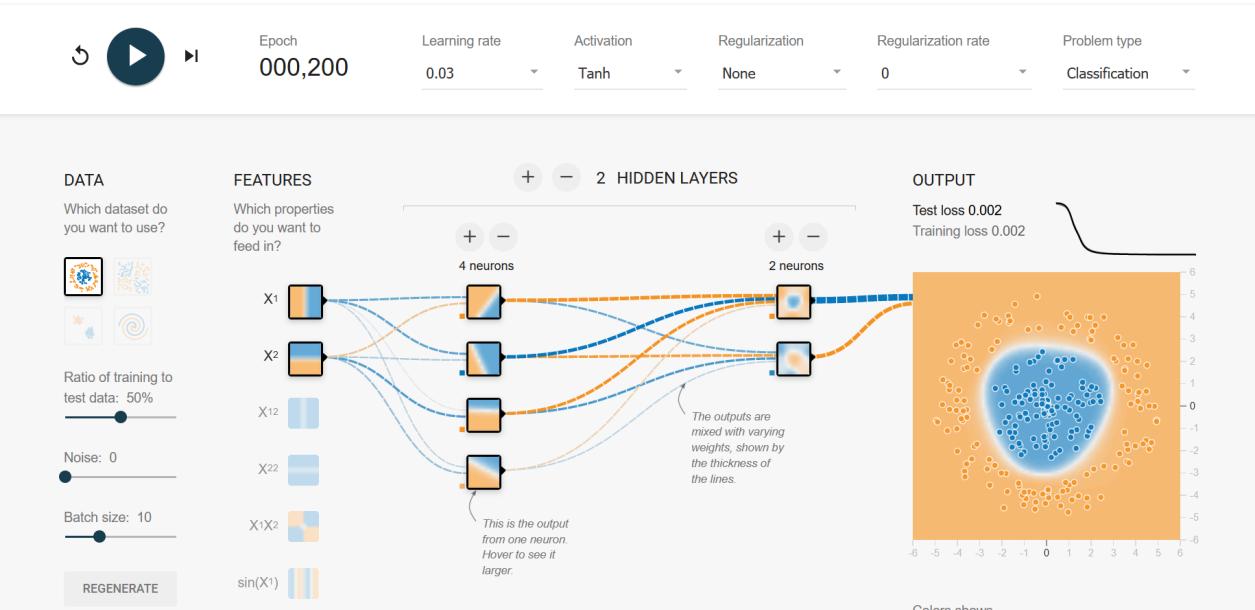
- $y = x^2$ のグラフは下に凸なので「中央と外」の区別がしやすいだろうと予想
 - 予想して選ぶならできて当然だろう！と思う人、その通りです
 - 画像認識AIも画像に特化して構成されている
- 膨大な計算量と時間を我慢するならば、工夫せずにやってよい
 - そこに金や時間を掛ける意味が????とならなければ、それでよい
 - 実際にはお金や時間を受けたくない人が殆ど
 - であれば、チートをしてでもはやすく解ける方法が欲しいと考えるのは当然
- このように、データの特徴を見出しその扱いになれているエキスパートがデータサイエンティスト
 - AIを勉強するというよりも、データサイエンティストとして、正しくデータを処理できる人材になるべきである、というのが言いたいこと



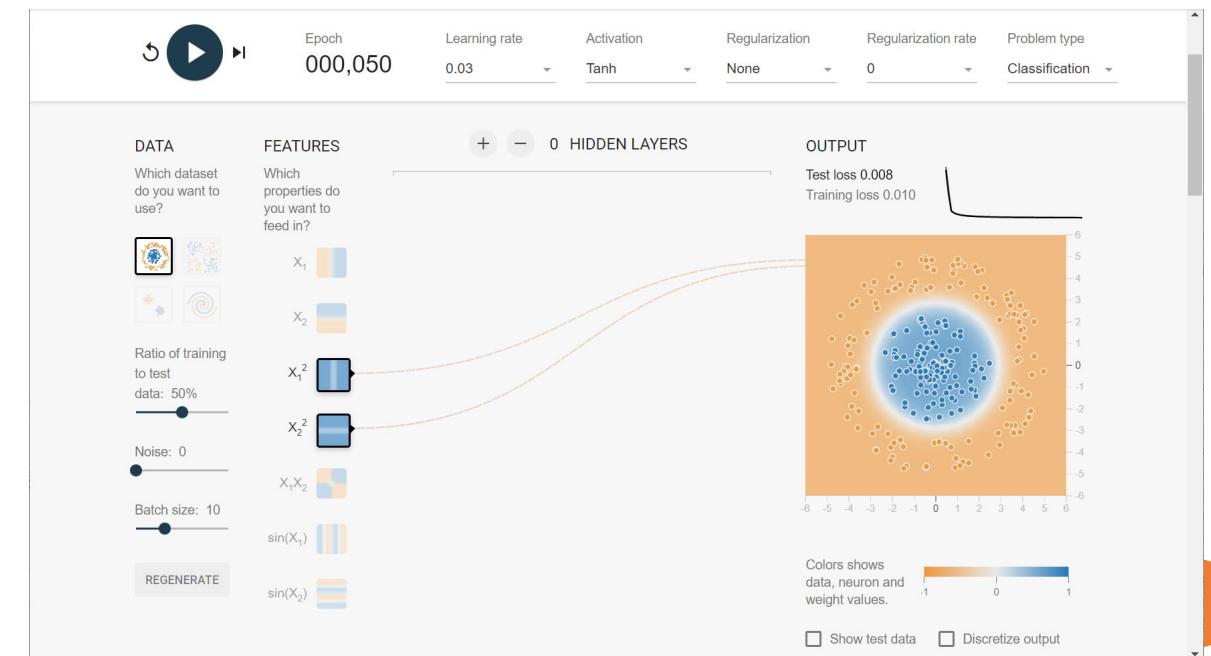
特徴量を工夫するとさらにいいことがある

34

- もっとシンプルにネットワークを構成できるかな？



できた！
中間層が2層で、 4×2 ノード
↓
中間層なしでいきなり出力
つまり、計算量がグンと下がるということ
安い計算機で学習できる！

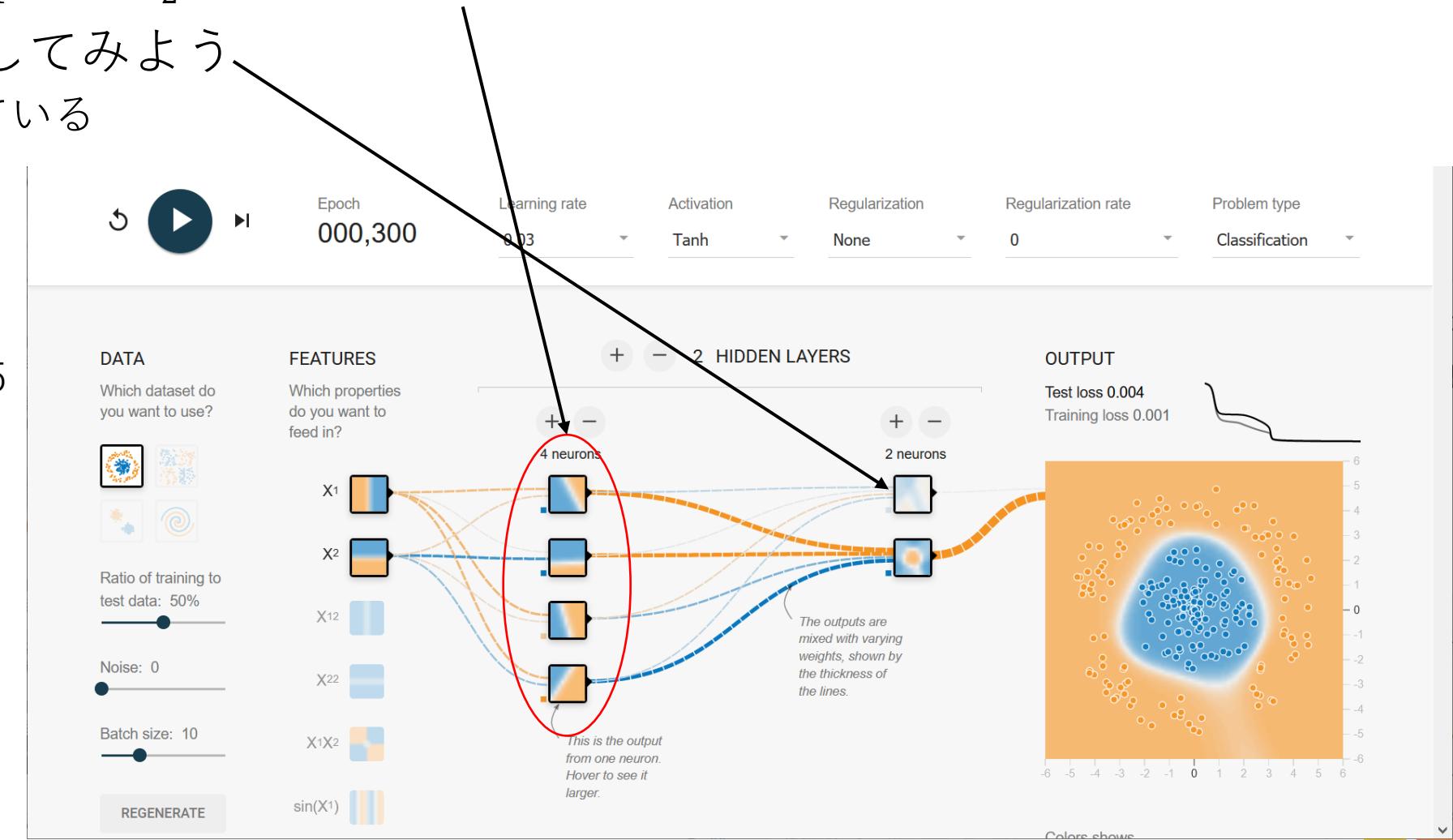




理解を深めさらなる工夫を考える

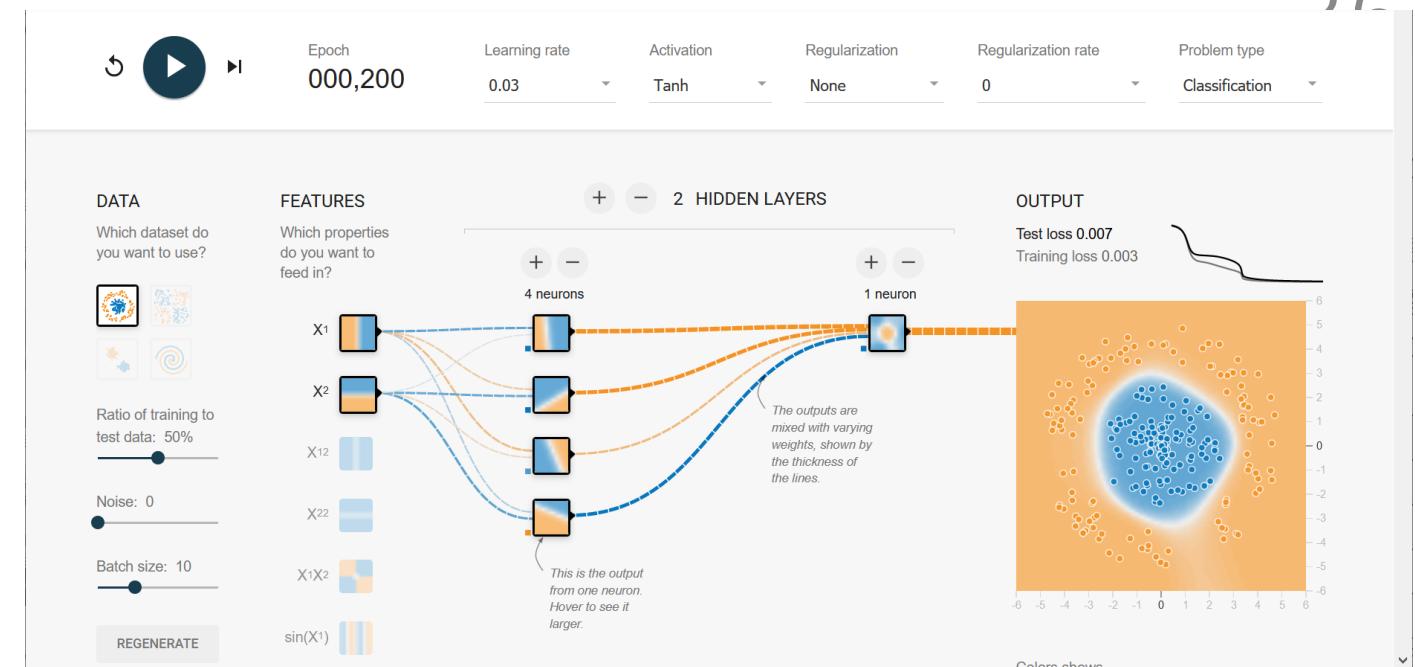
- なぜ、単純に温度と湿度それぞれについて $y = x$ という簡単な式を与えるだけで学習できたのだろうか？
 - 隠れ層を見ると、温度 x_1 と湿度 x_2 を足し合わせて斜めの線を描いている
- この部分は無駄？確認してみよう
 - 下だけでほぼ形ができている

$$\begin{array}{|c|c|c|} \hline -1 & 0 & 1 \\ \hline \end{array} + \begin{array}{|c|c|c|} \hline -1 \\ \hline 0 \\ \hline 1 \\ \hline \end{array} \times 0.5 = \begin{array}{|c|c|c|} \hline -1 & -0.5 & 0 \\ \hline -0.5 & 0 & 0.5 \\ \hline 0 & 0.5 & 1 \\ \hline \end{array}$$

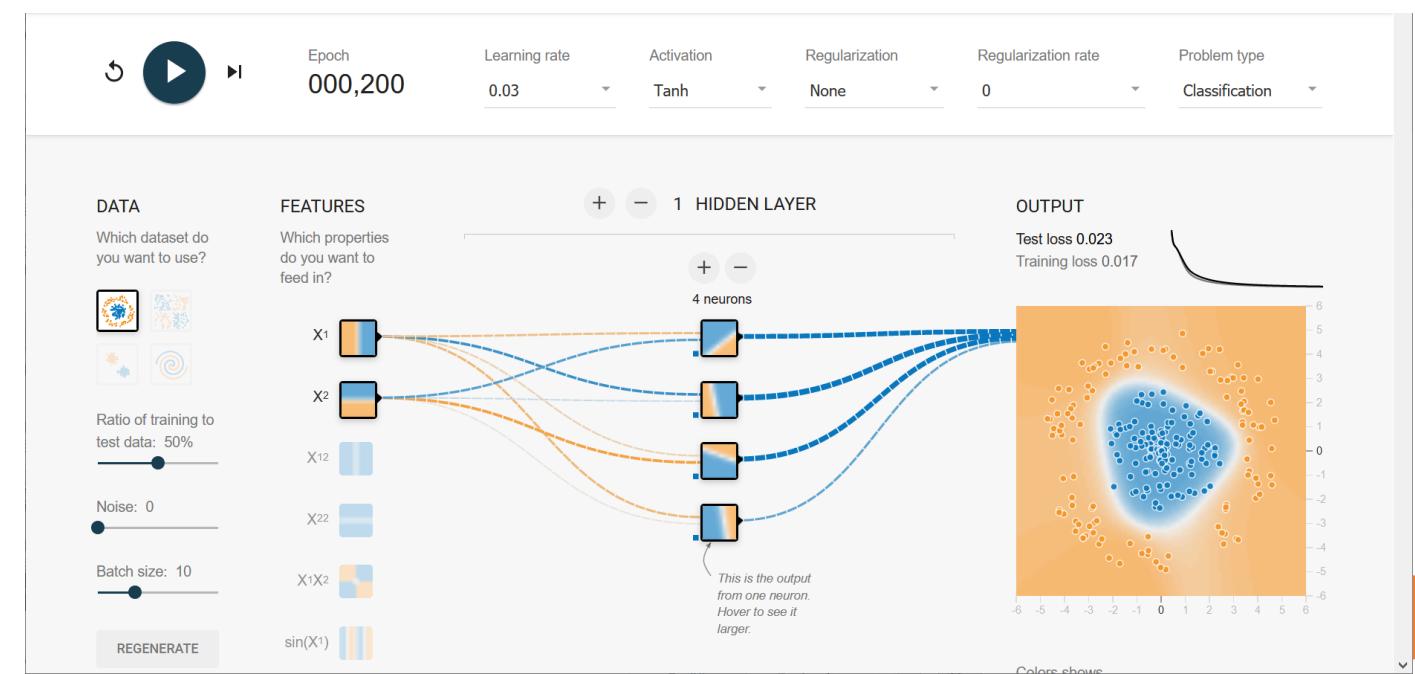
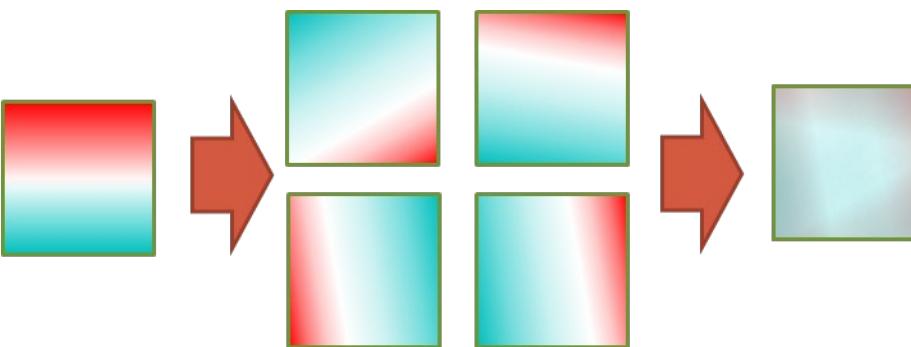


確かにできる

- 一つだけにした
- 案の定、中央が円で囲われた境界が現れている



- 実はなくてもよい
- 既に試した通り、最終形が隠れ層の内容であれば省略すればよい



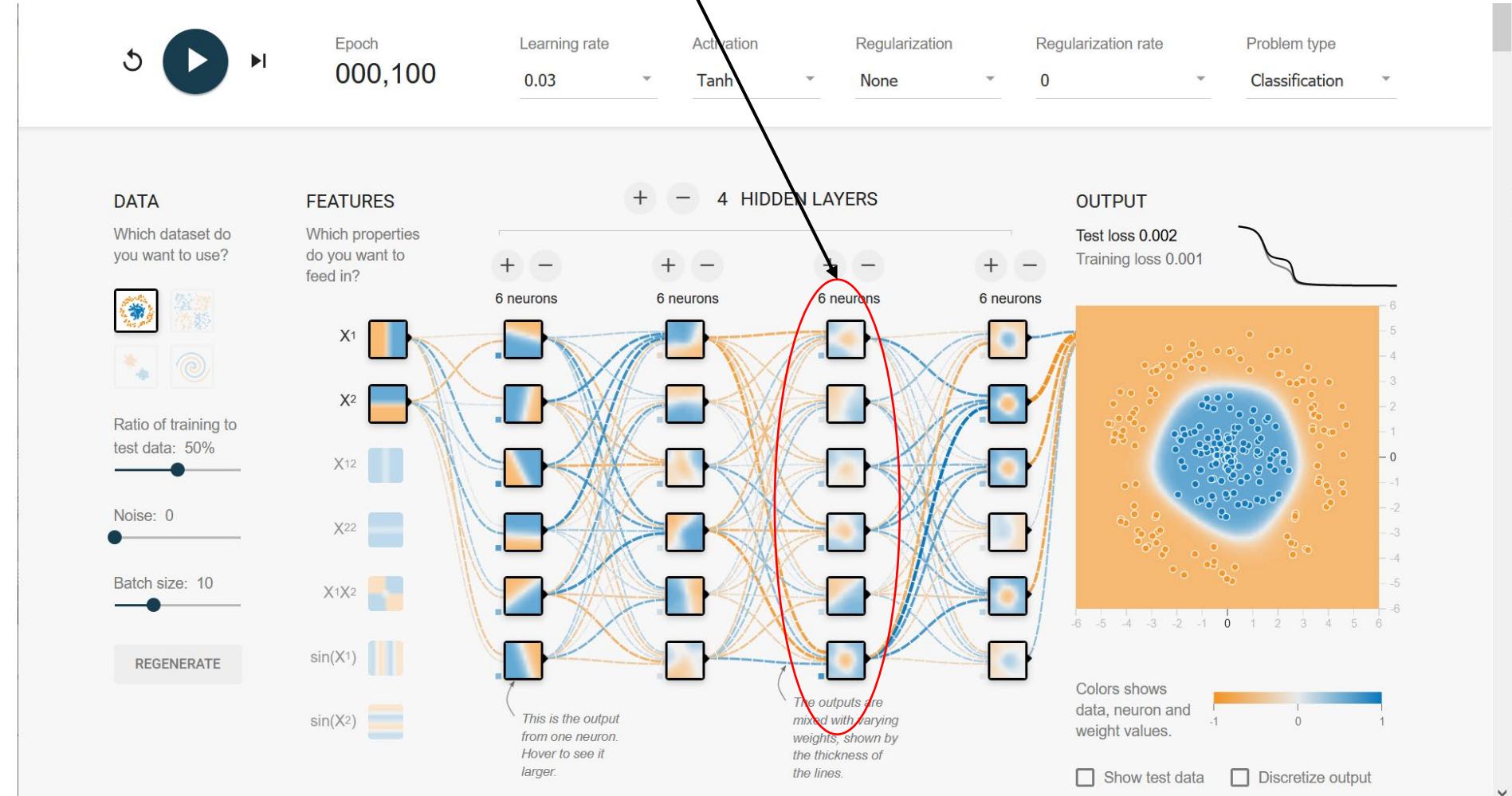
では、無駄に一杯使ってもできるのか？

37

- 何の問題もない

- つまり、「工夫しなくても十分な規模のネットワークを使えばなんでもできる」ということがDNNの重要なメリット

この辺りで既に欲しい分類ができている

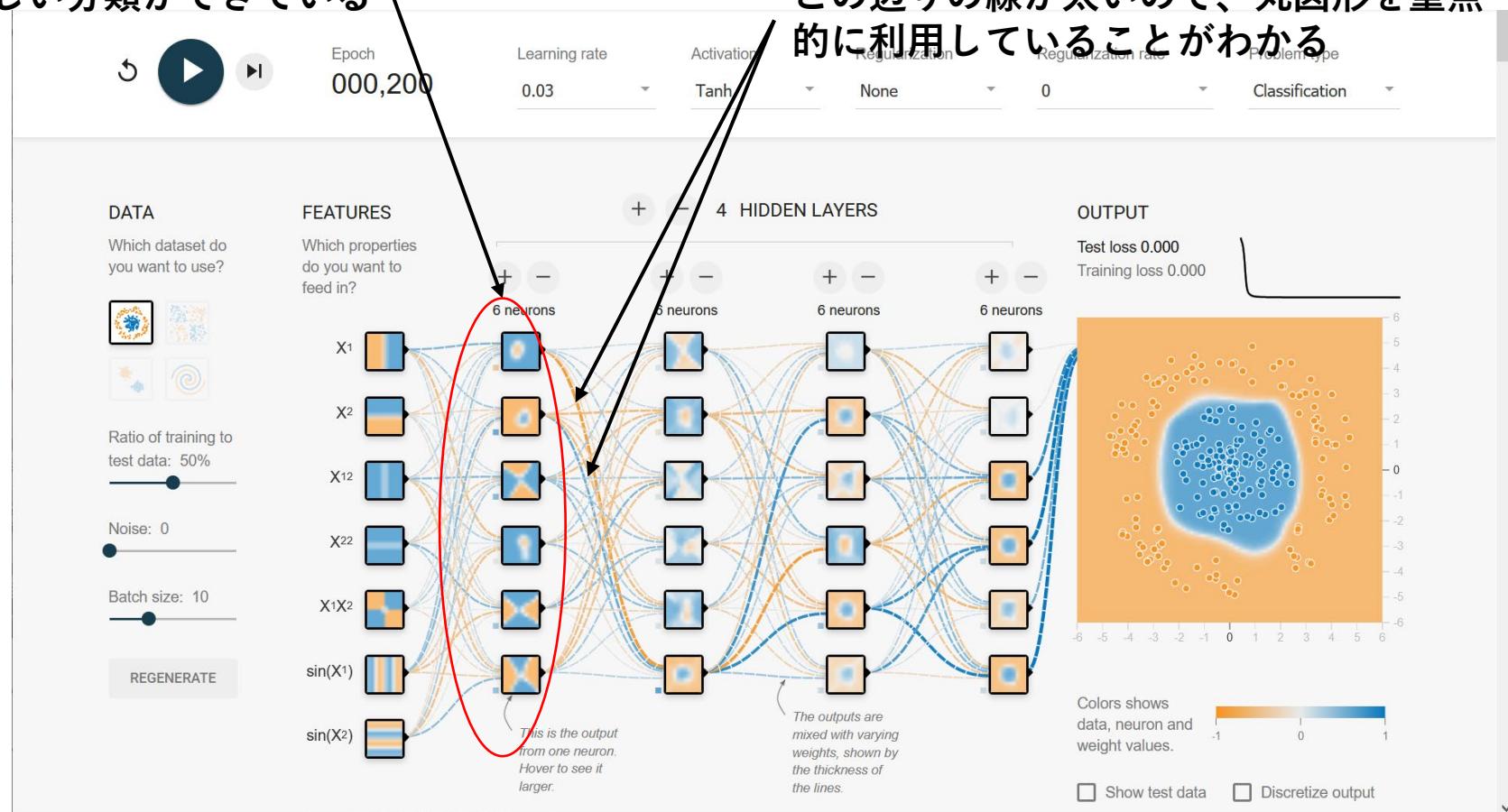


特徴量を一杯そろえてもできるのか？

38

- 何の問題もない
 - 途中で欲しい分類が得られていることから「無駄」であることはわかる
 - やはり、潤沢に特徴量やノードを準備しておけば、自動で学習するからうれしいであろう
 - これを自動というのならば、計算資源を潤沢に使って「自動」を実現すればよい

この辺りで既に欲しい分類ができている



実験してみよう

39

<https://playground.tensorflow.org>

The image shows the TensorFlow Playground interface with several annotations:

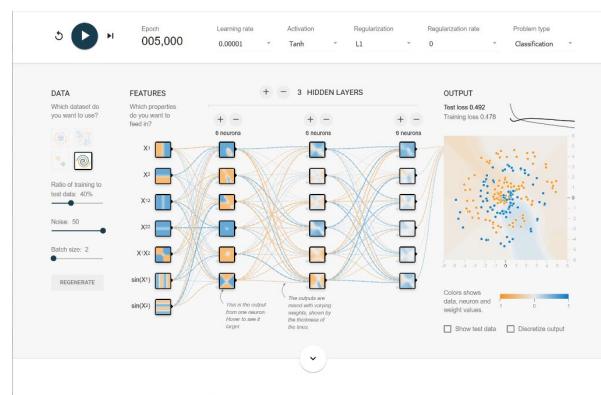
- 特徴量の選択**: Feature selection.
- 学習の実行**: Training execution.
- 学習のやり直し**: Re-training.
- 検証データの選択、決められたデータしか使えないのが残念だが、コードが公開されており変更できなくはない**: Validation data selection, regret about limited data choice.
- 全データを、学習用と検証用に分割するが、その割合**: Data split ratio.
- 値にノイズを入れてデータに変化を与える**: Noise addition.
- 与えられたデータを何件ずつに分けて計算し重みを更新するかのバッチサイズを決める**: Batch size decision.
- Epoch: 000,000**: Epoch count.
- Learning rate: 0.03**: Learning rate.
- Activation: Tanh**: Activation function.
- Regularization: None**: Regularization type.
- Regularization rate: 0**: Regularization rate.
- Problem type: Classification**: Problem type.
- 2 HIDDEN LAYERS**: Number of hidden layers.
- 4 neurons**: Neurons in the first hidden layer.
- 2 neurons**: Neurons in the second hidden layer.
- OUTPUT**: Output section showing Test loss 0.520 and Training loss 0.534.
- Colors shows data, neuron and weight values.**: Color scale for data, neuron, and weight values.
- 線の太さは重みの大小に比例**: Line thickness proportional to weight magnitude.
- 途中経過や結果の表示**: Intermediate process and results display.
- エポック = 試行回数**: An epoch is one full pass through the dataset.
- 1エポック = データセットサイズ ÷ (ミニ)バッチサイズ回のイテレーション**: One epoch is the number of iterations required to go through the entire dataset.
- 学習率：パラメータを勾配法によって更新するが、その更新量を調整する係数、初期値はそのままでよいが0.01などが良く用いられる**: Learning rate: A coefficient used to update parameters via gradient descent, often set to 0.01.
- 活性化関数**: Activation function.
- 隠れ層の数を変える**: Change the number of hidden layers.
- 過学習を避けるための正則化についてL1ノルムかL2ノルムを選択**: Choose L1 or L2 regularization to prevent overfitting.
- ノルムを足す割合、小さいと過学習大きいと学習不足**: Regularization strength.
- 回帰(Regression)と分類(Classification)を選択、今回は分類つまり、赤と青の領域分割をやってみる**: Choose Regression or Classification, this time classification.
- 損失関数の値、目標と実際の差の指標で、学習データの値と、検証データの値を表示**: Loss function value, target vs actual difference, training vs validation data values.

学習率を操作する

- 学習率が大きいときや小さいときはどうなるか？
 - 大きいと：得られた結果と望ましい目的値の差に対して大きく変化を与えるため、発散・振動しやすく収束しにくくなる
(答えを得るのに時間がかかる、もしくは答えに到達できない)
 - 小さいと：今度は変化が小さすぎて、発散・振動しない代わりに、収束が遅くなる
(やはり答えを得るのに時間がかかる)
- もちろん、様々な学習率の自動的設定手法が提案されている

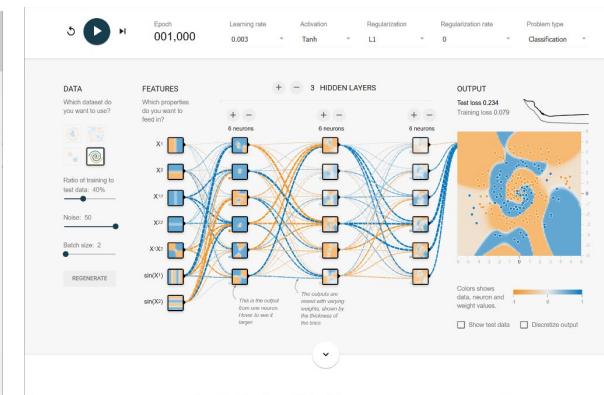
学習率=0.00001

エポック=5000でも途中



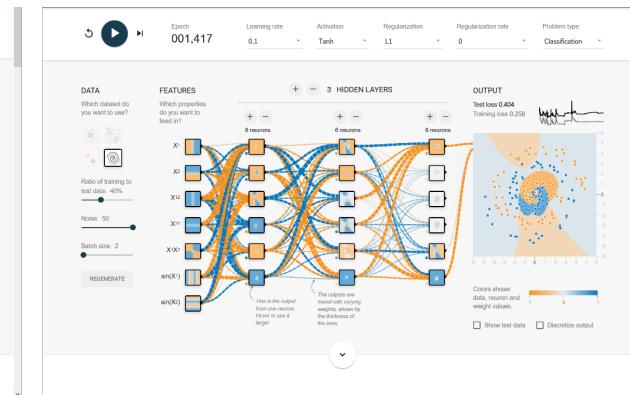
学習率=0.003

エポック=1000



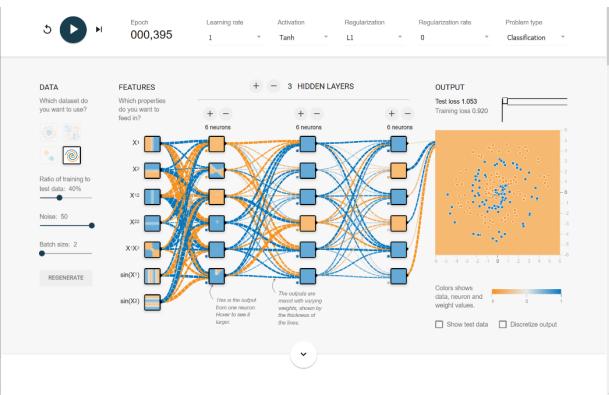
学習率=0.1

進展なし



学習率=1

全くダメ



正則化を操作する (1)

- 正則化が大きいときや小さいときはどうなるか？ (L1もしくはL2を選択しておく)
 - この場合、複雑な渦巻きなど、乱雑な方がわかりやすい
 - 正則化がないと振動しがちで収束が遅く、値が大きいと、そもそも収束しない

L1=0.001

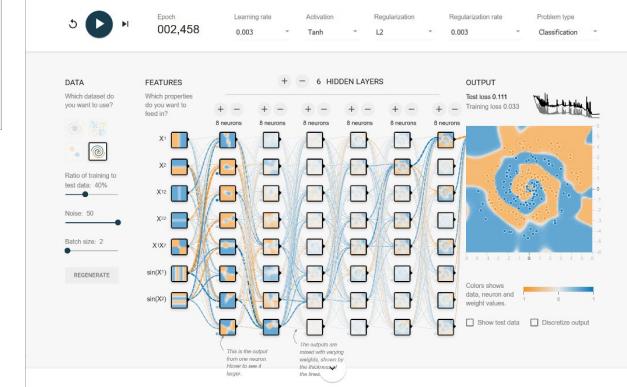
なし



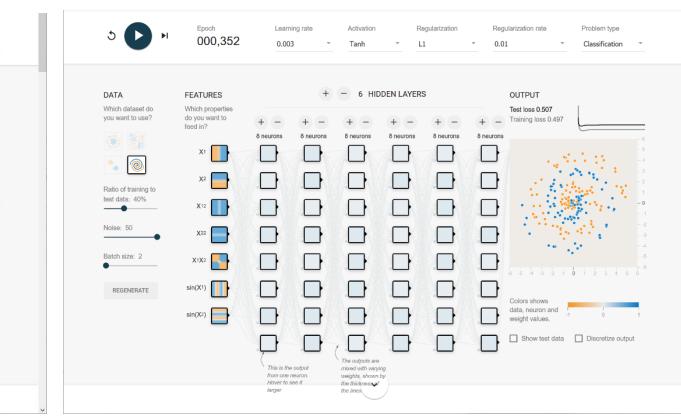
L1=0.003



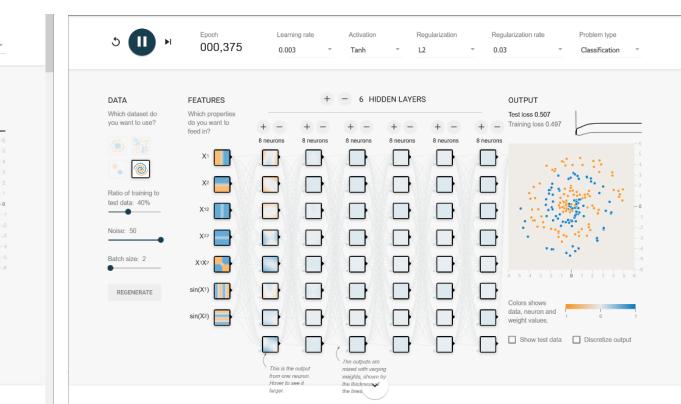
L2=0.003



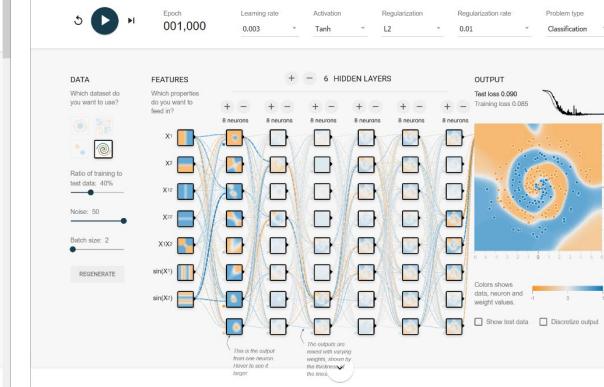
L1=0.01(失敗)



L1=0.03(失敗)



L2=0.01

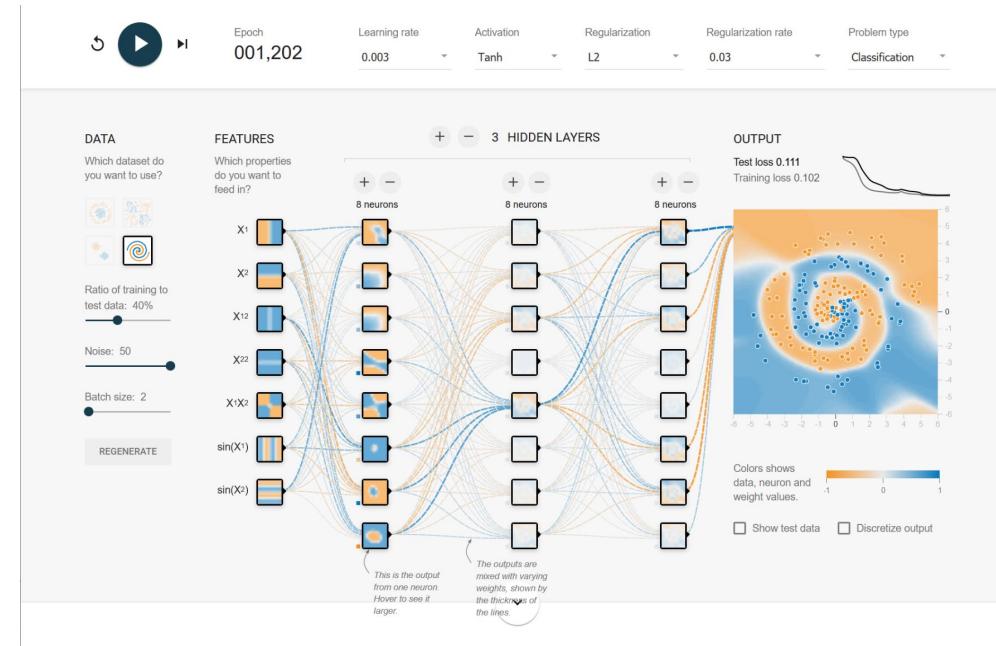




いろいろやってみよう！（正則化2）

42

- 層数が異なるとき、正則化が大きいときや小さいときはどうなるか？
(L1もしくはL2を選択しておく)
 - 層数が大きいほど、小さい値で発散する



- L1正則化は主に余分な説明変数を省くために用いられる(w_i が0になりやすくする)
- L2正則化は過学習を防ぎ精度を高めるために用いられる(L1正則化よりも予測精度が高い傾向にあり、また、モデルのパラメータがより滑らかかつ表現力に優れるため)

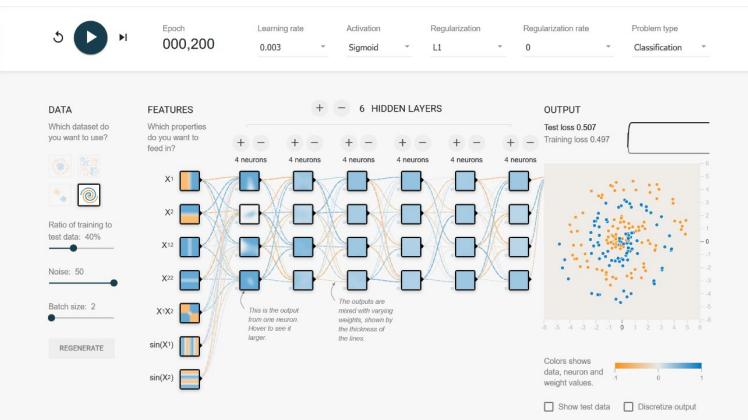


なぜ多層NNの学習が難しかったのかを確認

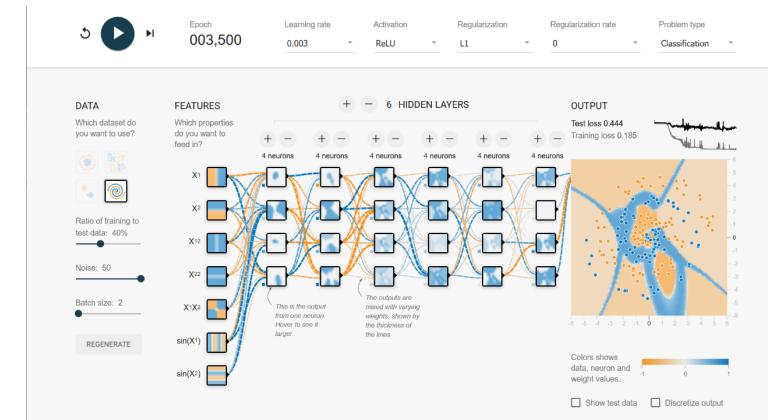
43

- 層数が大きい場合、活性化関数にシグモイドを選ぶと収束しない
 - DNNではシグモイドが使えない。ReLUの発明がDNNの世界を切り開いた
 - 一方で、隠れ層が多いから良い結果ができるというわけではないことも確認する

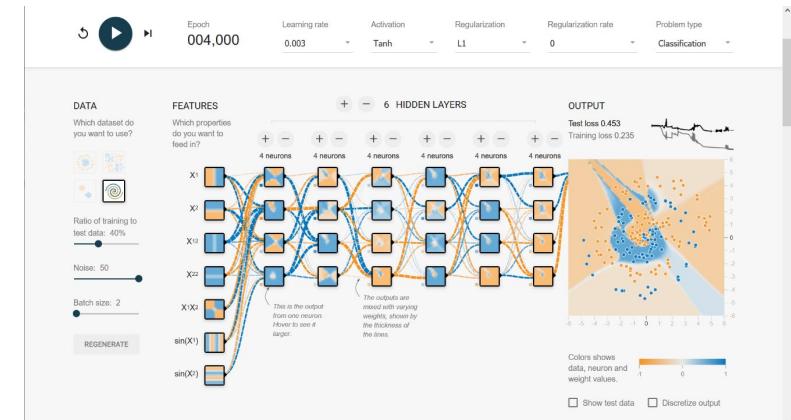
シグモイド(ダメ)



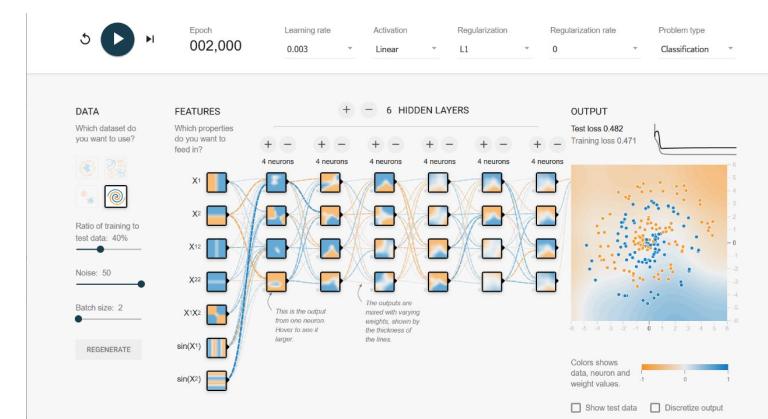
ReLU



tanh



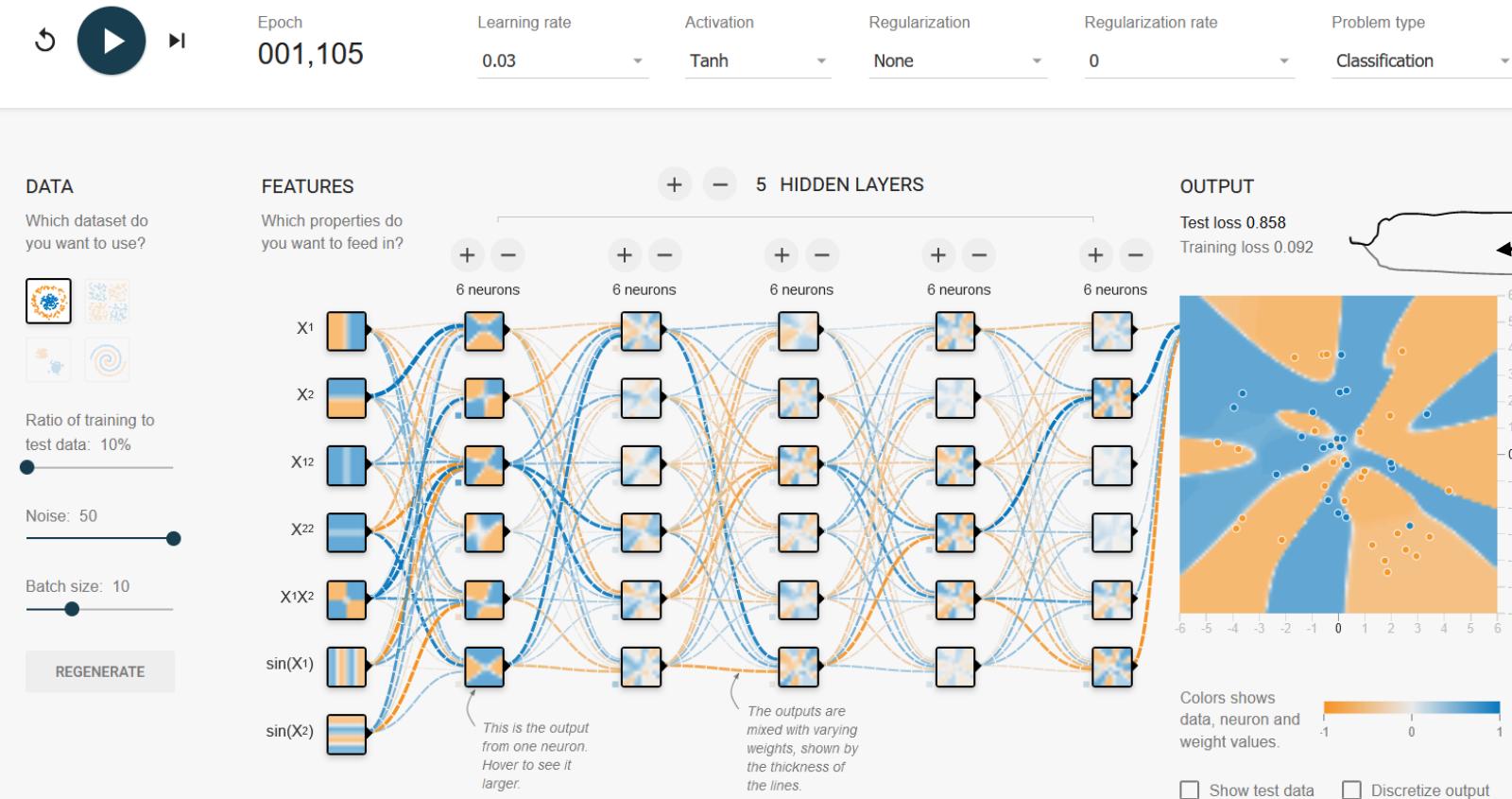
リニア($y=x$ 、分類には不向き)



過学習ってなに？

44

- ・何事もやりすぎてはダメ！
- ・時間をかけすぎると、とにかく頑張って「細かく」区分けしようとする
- ・グラフで学習データは値がよくなる(小さくなる)が、検証データは悪くなる



- ・学習用データは値がよくなる
 - ・検証用データは値が悪くなる
 - ・これがダメ！
- バランスを見よう！これが過学習を抑える分類結果が欲しかったわけではない！



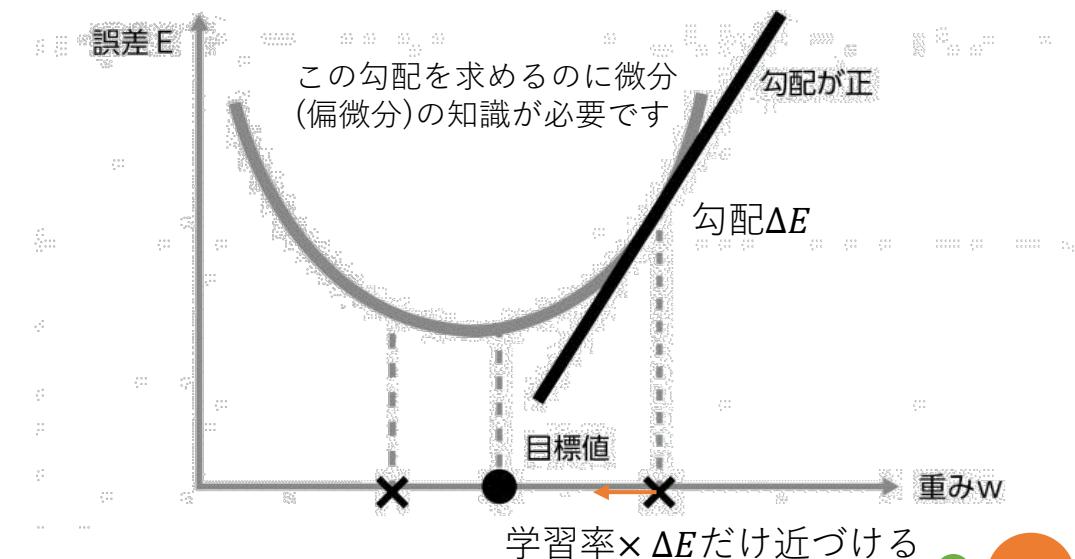
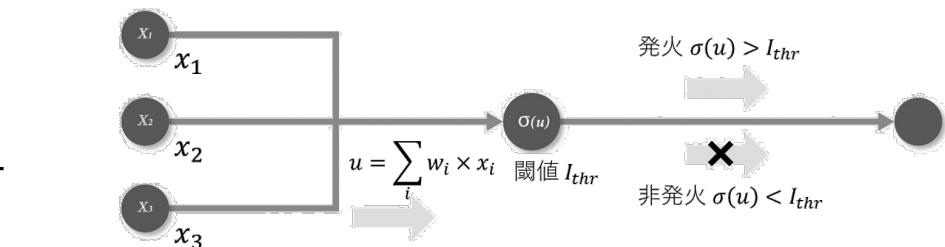




どうして学習できるのか？

46

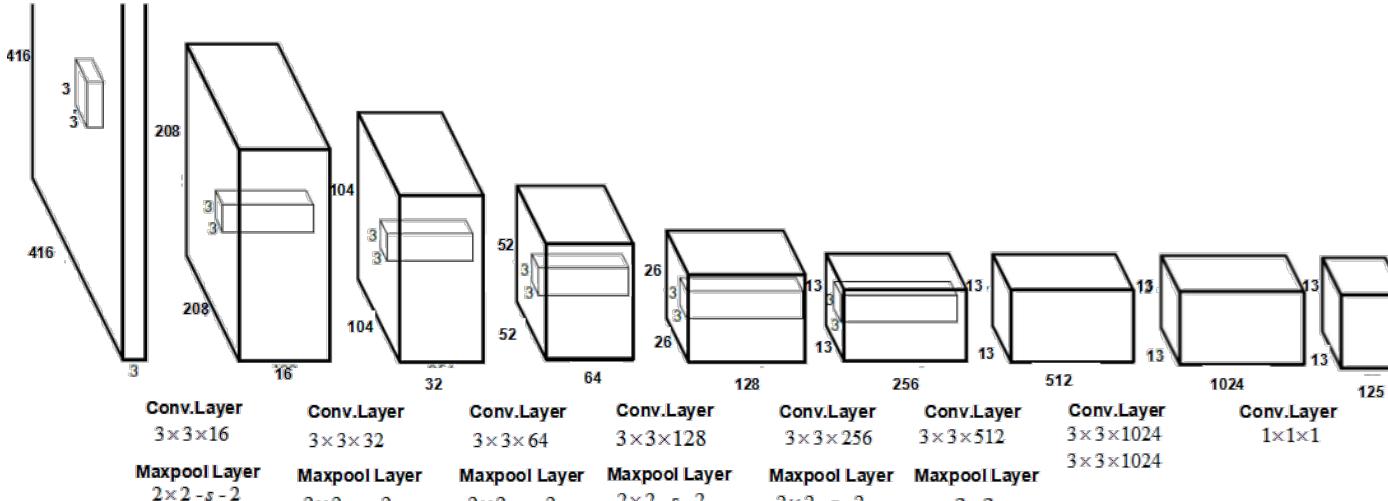
- まず、シンプルなネットワークについて考える
- 学習とは、教師データを用いてネットワークの重み調整を行うことである
 - 教師データは、入力とその正解のセットで構成される
 - つまり学習とは、右図の w_i の値を調整することである
- シンプルなネットワークでは次の方針が取られます
 - 重みとバイアスを初期化する（これは適当で構わない）
 - データ（バッチ）をニューラルネットワークに入力し出力を得る
 - この出力と正解の誤差を損失関数で計算
 - 誤差をへらすように重みを修正
 - 正しいといえる重みになるまで、2から4を繰り返す
 - これを勾配降下法と呼ぶ
 - 近づけるときにどれだけ寄せるか？が「学習率」です
- 一発で正解値を出すのは複雑すぎてほぼ無理
 - そこで次第に正しい値に近づける方針を取ります



例えば、シンプルな画像認識でも…

47

- では、どの程度多層なのか？どの程度計算が大変なのかをみてみる
- シンプルな物体認識DNNとして知られるTiny-YoLoのネットワークは次の通り
 - 例えば、(ここでは厳密さは省き、詳細は改めて説明する)416x3は、カラー画像を416x416ピクセルx3の情報とし、これを208x208x16個のノードへ畳み込み、次に104x104x32のノードと畳み込みといった具合に、全23層を情報が伝わるように構成されている
 - 大量にノードが存在しており、これらの重みを全て安定させ定常状態になるようにする必要がある
 - 「まじか—————」と思いますよね。現実はもっと、もっと多いです
 - こんなものを安定的に学習させる技術ができてしまった
 - これがDNNの発展を推し進めた一番の理由



Layer	Type	Filters	Size/Stride	Input	Output
0	Convolutional	16	3x3/1	416x416x3	416x416x16
1	Maxpool		2x2/2	416x416x16	208x208x16
2	Convolutional	32	3x3/1	208x208x16	208x208x32
3	Maxpool		2x2/2	208x208x32	104x104x32
4	Convolutional	64	3x3/1	104x104x32	104x104x64
5	Maxpool		2x2/2	104x104x64	52x52x64
6	Convolutional	128	3x3/1	52x52x64	52x52x128
7	Maxpool		2x2/2	52x52x128	26x26x128
8	Convolutional	256	3x3/1	26x26x128	26x26x256
9	Maxpool		2x2/2	26x26x256	13x13x256
10	Convolutional	512	3x3/1	13x13x256	13x13x512
11	Maxpool		2x2/1	13x13x512	13x13x512
12	Convolutional	1024	3x3/1	13x13x512	13x13x1024
13	Convolutional	256	1x1/1	13x13x1024	13x13x256
14	Convolutional	512	3x3/1	13x13x256	13x13x512
15	Convolutional	255	1x1/1	13x13x512	13x13x255
16	YOLO				
17	Route 13				
18	Convolutional	128	1x1/1	13x13x256	13x13x128
19	Up-sampling		2x2/1	13x13x128	26x26x128
20	Route 19 8				
21	Convolutional	256	3x3/1	13x13x384	13x13x256
22	Convolutional	255	1x1/1	13x13x256	13x13x256
23	YOLO				



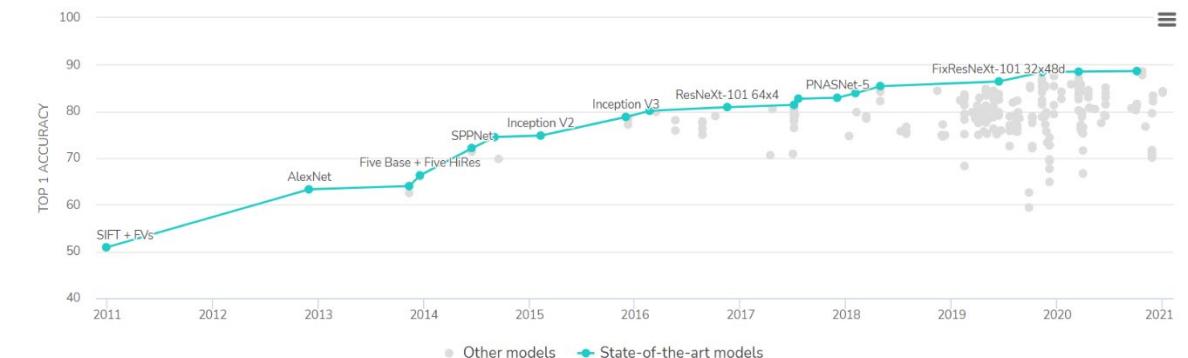


ここまでまとめ

48

- ここ5年のDLブームは本物
 - 特にDLはすごいが何が凄いのか？
 - 端的には、パラメータを山のように増やしても過学習がなぜか起きないこと
 - これまでの常識がすべて覆った
 - 説明可能とかくだらないことに拘るのは年寄り
 - 人間ならよいと言い出すが、人間は説明不可能
- この領域は日進月歩
 - 物体識別精度の記録は日々更新
 - 今学んでいる内容も将来メジャーとは言えない
- 自由度は高い
 - 一方で「大概モデル」も相当数存在
 - 「先人の知恵を学ぶ」ことに主眼を置く
- 一方で
 - 世界の頭脳と金がモデル提案にまい進している現時点で勝負を挑むなら相当覚悟した方が良い

Image Classification on ImageNet



View	Top 1 Accuracy	All models							
RANK	MODEL	TOP 1 ACCURACY	TOP 5 ACCURACY	NUMBER OF PARAMS	EXTRA TRAINING DATA	PAPER	CODE	RESULT	YEAR
1	EfficientNet-L2-475 + SAM	88.61%		480M	✓	Sharpness-Aware Minimization for Efficiently Improving Generalization	View	Edit	2020
2	ViT-H/14	88.55%		632M	✓	An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale	View	Edit	2020
3	FixEfficientNet-L2	88.5%	98.7%	480M	✓	Fixing the train-test resolution discrepancy: FixEfficientNet	View	Edit	2020
4	NoisyStudent (EfficientNet-L2)	88.4%	98.7%	480M	✓	Self-training with Noisy Student improves ImageNet classification	View	Edit	2019
5	ViT-L/16	87.76%		307M	✓	An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale	View	Edit	2020

