

自然言語処理 Deep Learning

—導入—

<https://satoyoshiharu.github.io/nlp/>

導入の概要

授業の目的

授業の概要

環境準備

この授業の目的は？

3つのゴールから選択

Pythonをマスターする

機械学習・ニューラルネット
を理解し、自分の課題
に取り組めるようになる

自然言語処理アプリを
組めるようになる

導入ヒヤリング 1

過去に機械学習、ニューラルネットワーク、あるいはディープラーニングの授業をとったか、自分で本を勉強したことがある人は？

人工知能って何に使えるの？

制御、診断

- 異常検知
 - 工場の製造部品や、野外の建造物などを、カメラ等でモニターし、不良品や異常を検知する。
- ロボット制御
 - カメラやセンサーの入力信号から、次の行動を決定し、モーター駆動機構へ命令する。
 - 自動運転。
- 病理診断
 - レントゲン写真から病気を見つける。

予測

- 株価予測
 - 過去のデータから、今後の株価を予測し、売買を実行する。
- レコメンデーション
 - あるユーザの過去の購買履歴やブラウザ履歴から、その人の関心を予測し、関連する広告を選んで表示する。
- 顧客属性予測、行動予測
 - あるユーザの過去の購買履歴や投稿内容から、優良顧客かクレームユーザになりそうかを予測する。

情報変換・加工（して人を助ける）

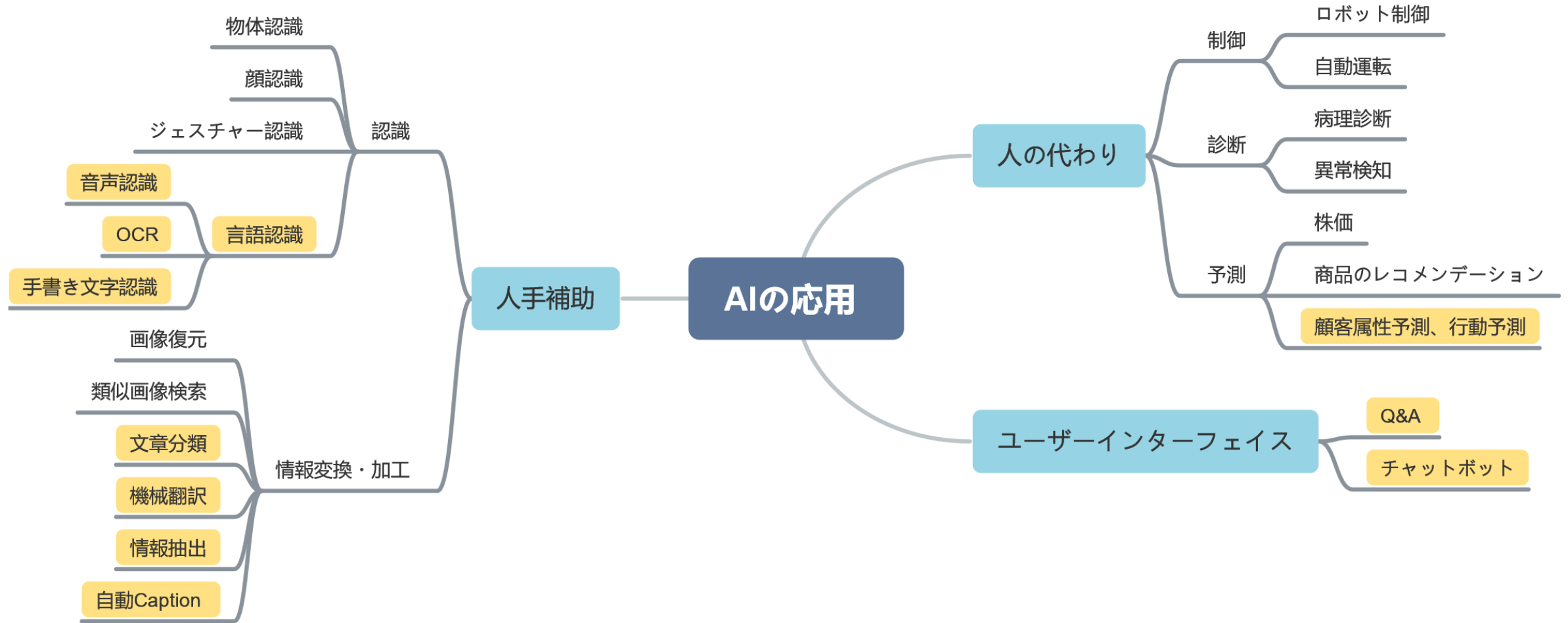
- 画像復元
 - 人工衛星の地表映像から、より鮮明な地表映像を復元する。
- 類似画像検索
 - 指定された画像と類似したものを、提示する。
- 文章の分類
 - 文章を読み、カテゴリーに分ける（例：商品コメントを、ポジティブとネガティブに分けて、ユーザの評価を客観視する）。
 - 検索したページに含まれる記事を読み、カテゴリーに分類し、キュレーションサイトとして提供する。
- 機械翻訳
 - ある言語の文章を、他言語に翻訳する。
- 情報抽出
 - 商品へのコメントから、重要なキーワードを抽出し、評判を分析する。
- 画像キャプション
 - 画像に、表現されている物体を文章化する。

認識（して後続の処理に提供する）

- 物体認識
 - カメラ画像から、物体を認識する（例：自動車の車載カメラで、対向車や横断歩行者を認識）。
 - 手指を認識し、空中のジェスチャーを認識する。
- 顔認識
 - ある人の顔と、他の人の顔と区別し、個人を特定する。
- ジェスチャー認識
 - 加速度センサー、ジャイロ스코プセンサーの信号から、装置の動きを計算し、装着者のジェスチャーを認識する。
- 音声認識
 - マイク入力音声信号から、話された文章を認識する（例：Siri、Echo）。
- OCR
 - 印刷された文字を認識し、文字コードに復元する。
 - 外国で、メニューをカメラに映して、自国語に翻訳させる。
- 手書き文字認識
 - 手書き文字を認識する。

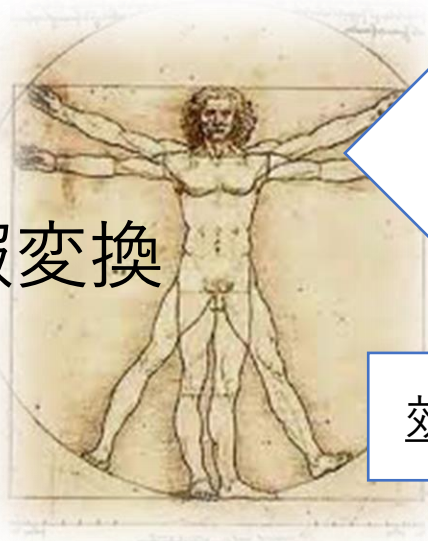
インターフェイス

- Q&A
 - サポートへの質問に対し、回答を生成する。
- チャットボット
 - ユーザのタイプした文章に、適当な回答をして、対話する。



人は膨大な量の情報変換をしている。人工知能はそれを模倣する。

情報変換



人

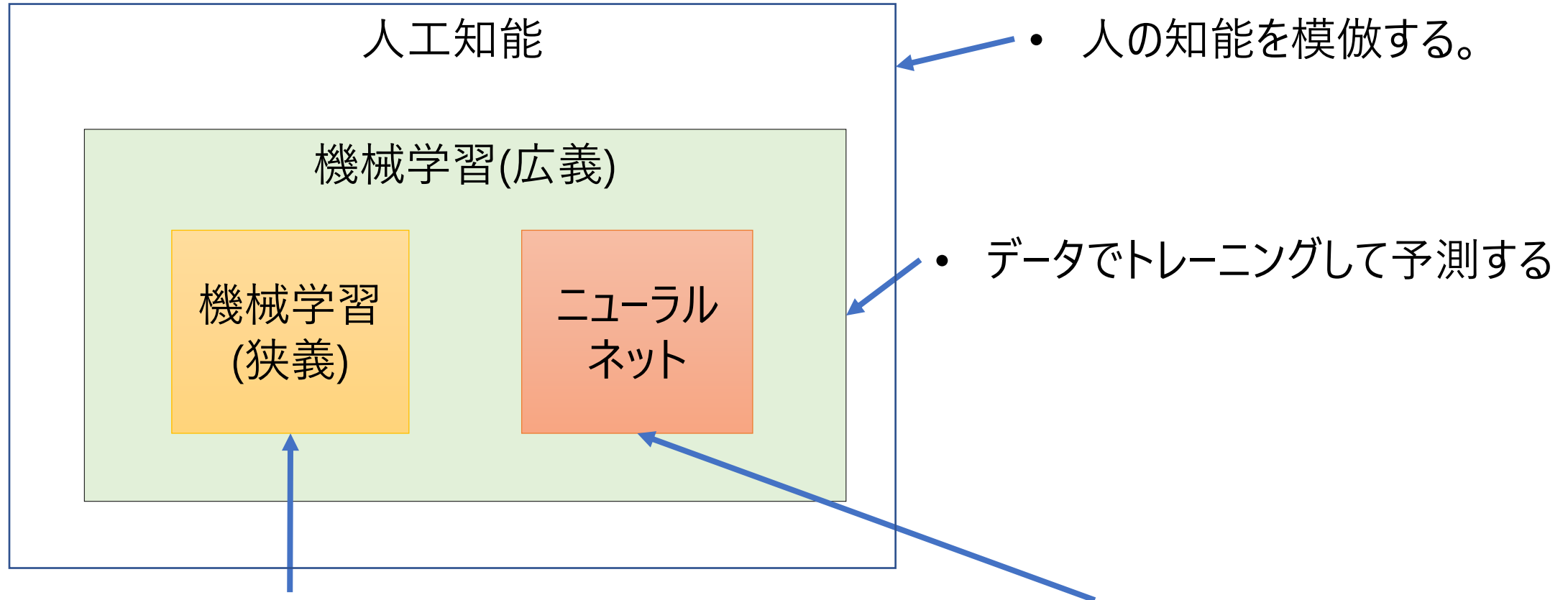
受容器官: 10億ビット/秒

効果器官: 1千万ビット/秒

環境



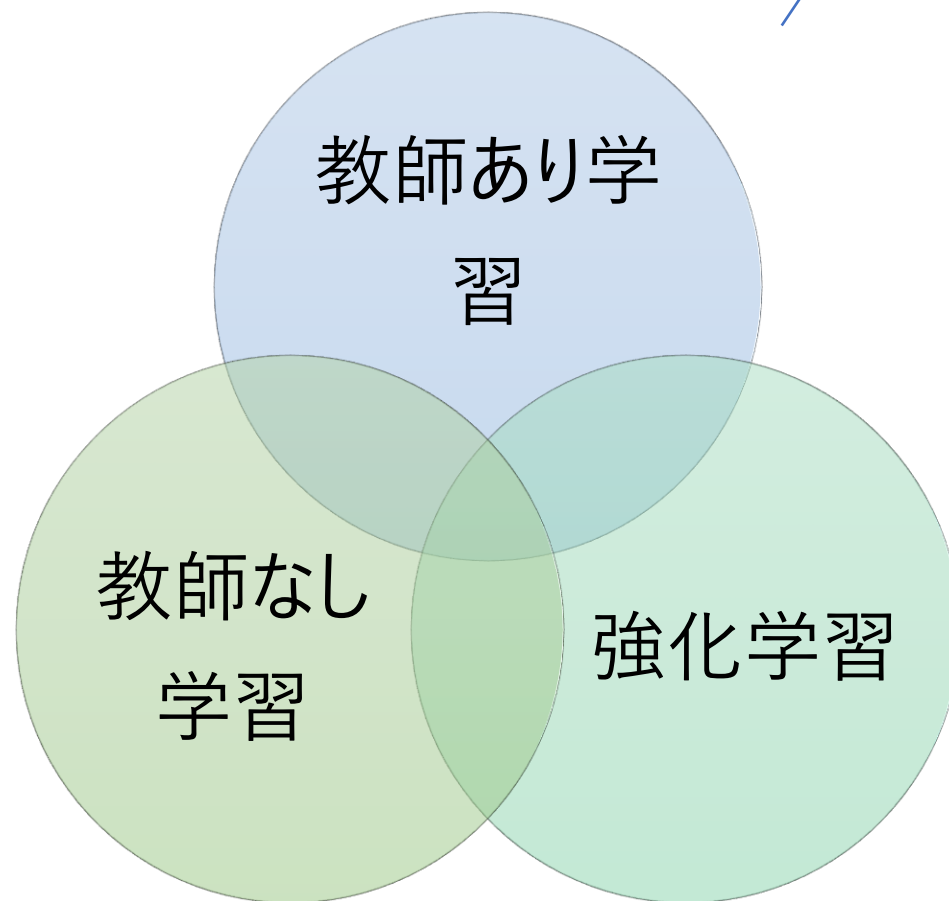
人工知能、機械学習、ニューラルネットの違い



- 推論根拠となる入力特徴を人が定義する
- 推論根拠がわかりやすい

- 推論根拠となる入力特徴を機械が学習する（複雑な対応関係でもOK）
- 推論根拠がわかりにくい

学習の観点からの分類

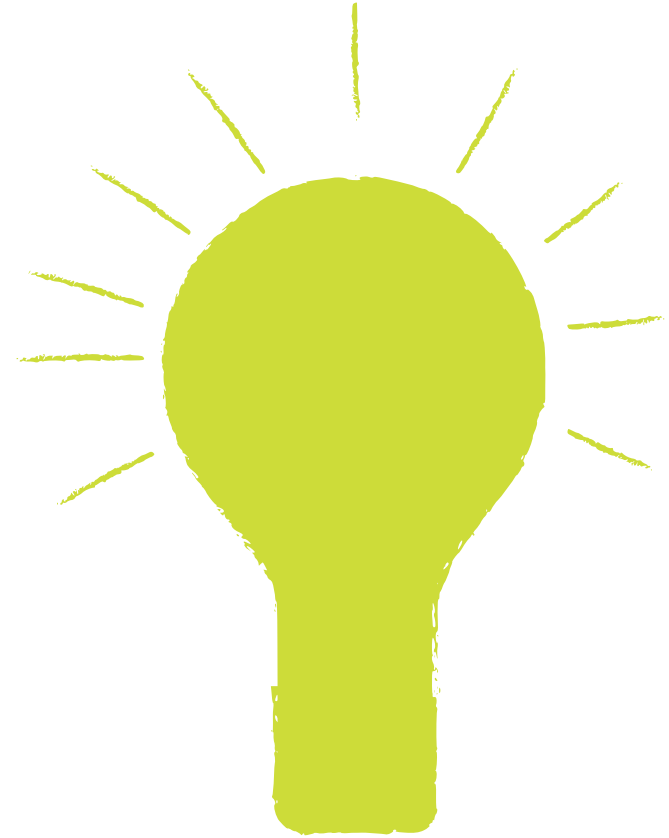


認識、変換、診断、
予測

ゲーム、
制御

A I の応用 = 人がやっているような情報変換を
模倣したもの

応用は無数にありうる。
その限界はあなたの想像力のみ。

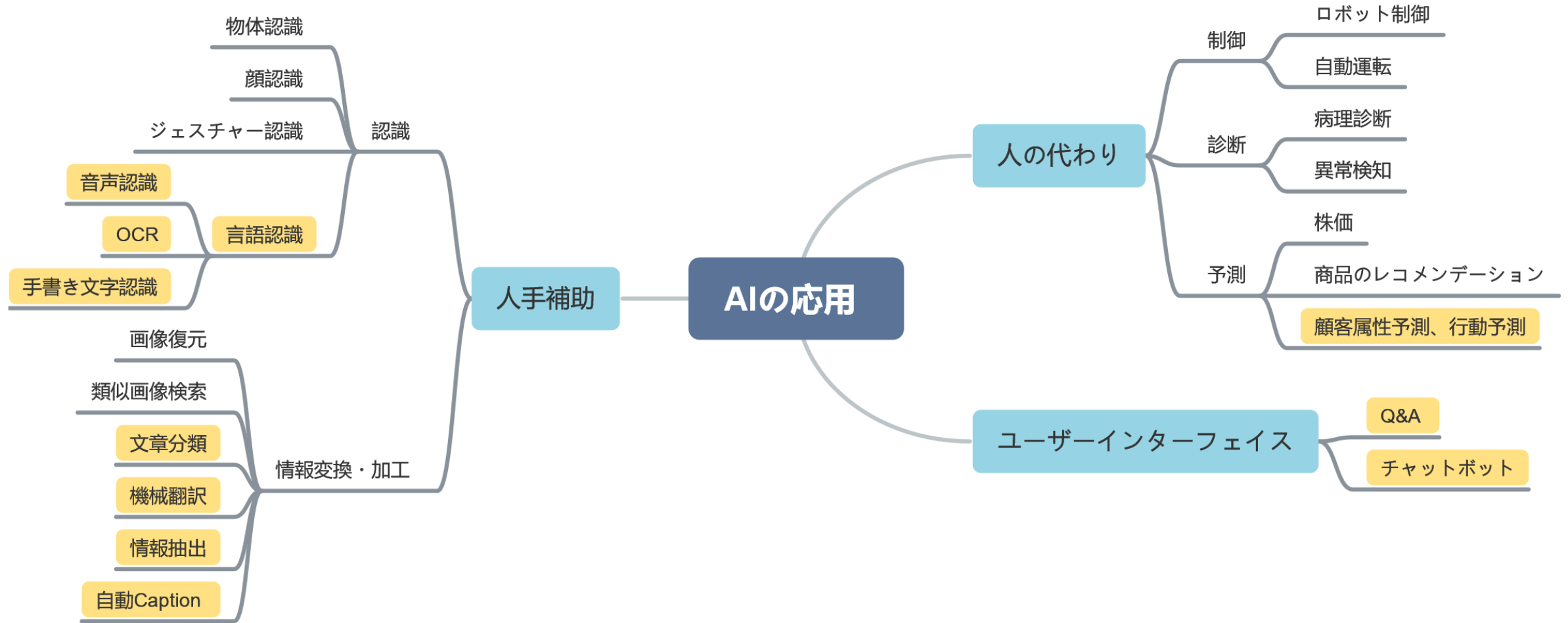


参考

- [ディスカバリーチャンネル、AI革命～激変する世界を探る～Part1 言語の理解/機械との議論/路上を走行する人工知能](#)
- [Deep Learning入門：Deep Learningでできること](#)
- [Deep Learning入門：今Deep Learningに取り組むべき理由](#)
- [自然言語処理 05 深層学習×自然言語処理](#)
- [Deep Learningで行う自然言語処理入門](#)

自然言語処理って？

先の事例の中で、信号・画像でなくて、
人がしゃべったり、書いたりした言葉を
対象にしたものは？



センサーの信号処理、画像処理と比べて、自然言語処理のDeep Learningは、どう違う？

連続量と離散量

- 連続量

- 例

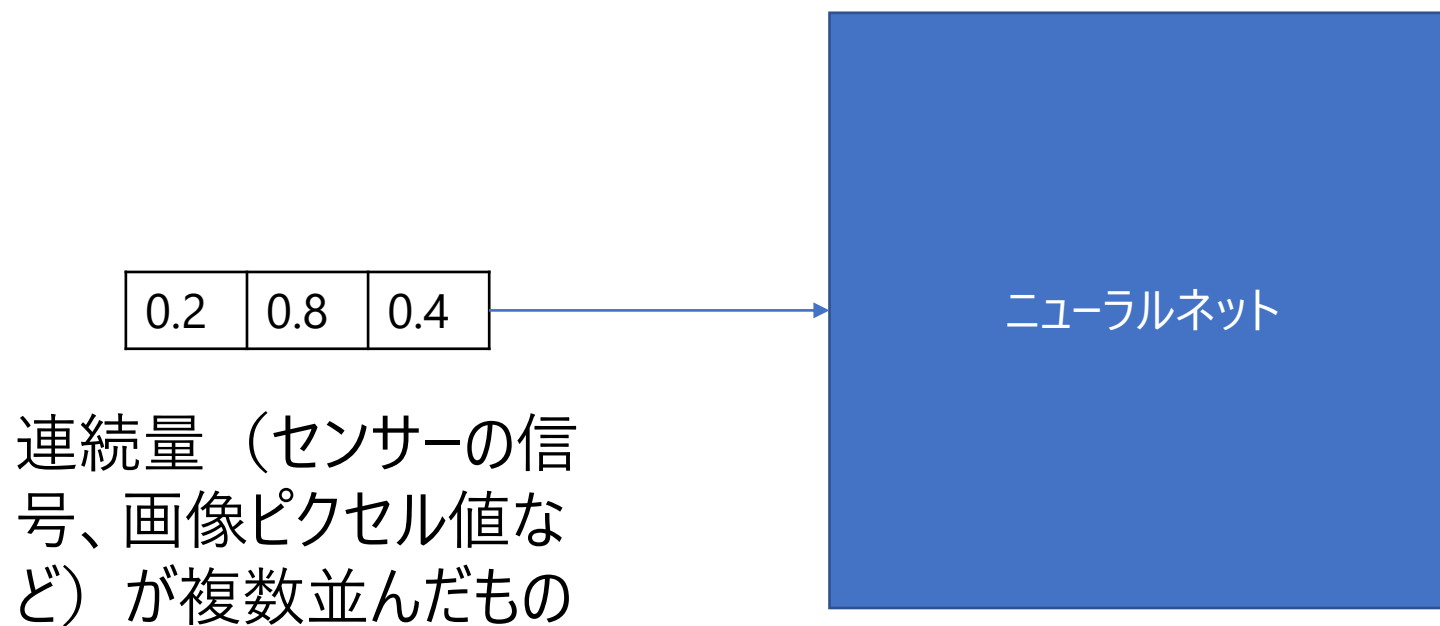
- 身長：170cmと170.11cmとの間に無限に取りうる値がある。
 - 体重
 - カメラから写っている物体までの距離
 - 画像のあるピクセルのRedチャンネルの量(floatの場合)
 - 加速度センサーのX軸の反応量
 - スピーカーの音量
 - ...

- 離散量

- 例

- さいころの目：1から6の間の値を飛び飛びでとり、それらの間の値を取りえない。
 - 何月か
 - 学生番号
 - トランプのマーク
 - 人の性別
 - コロナ検査で陽性か陰性か
 - ...

ニューラルネットの入力は通常連続量のベクトル



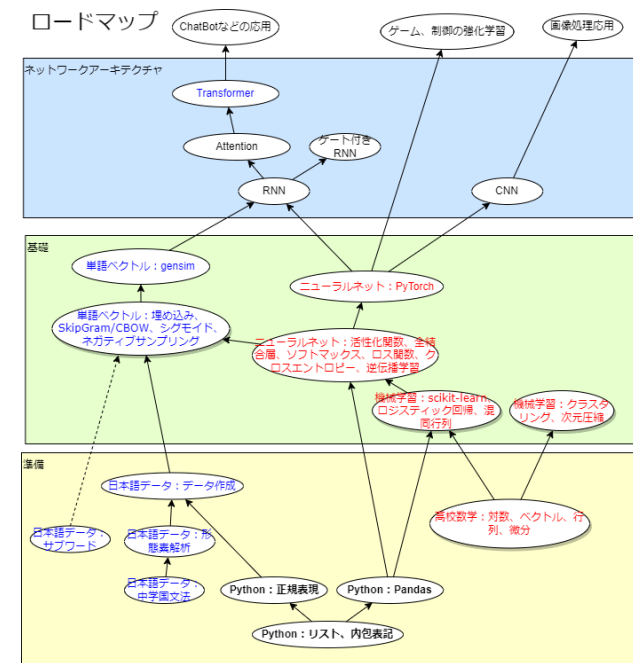
自然言語の処理対象は離散的なので、データの変換が必要

- 文字：あ、い、山、川、...
- 単語：今日、明日、太郎、花子、...
- 文：今日は晴天です。明日は木曜日です。週末が楽しみです。
...

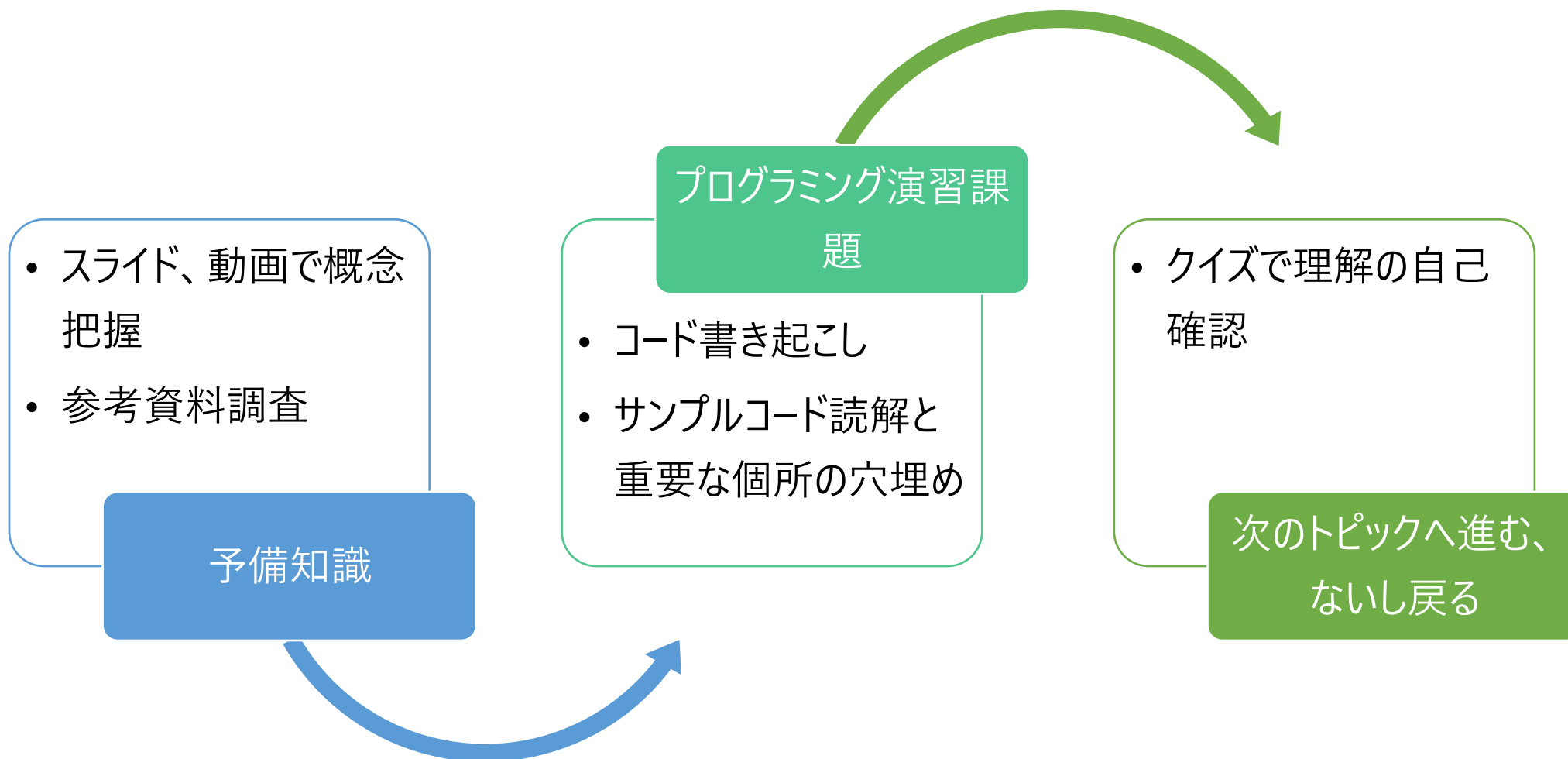
この授業で何をする？

学習ロードマップ

- 高校数学の復習、Pythonの復習から、機械学習、Deep Learningと広範囲で、かつ最先端の高度なトピックまでカバーしますが、どこをどう攻めるかは選択式です。
- NLP学習ロードマップ.png をご覧ください。



この授業で何をする？



教科書はありません

- いい教科書がない。
 - 既存の教科書は、大学の教養課程の数学（線形代数、解析）を前提としていて、いきなり難しい数式が出てきて説明がない。
 - この分野は変化が激しいため、本になったものは、たいてい、書籍になった時点で、実は、扱う内容が、すでに時代遅れです。

= >

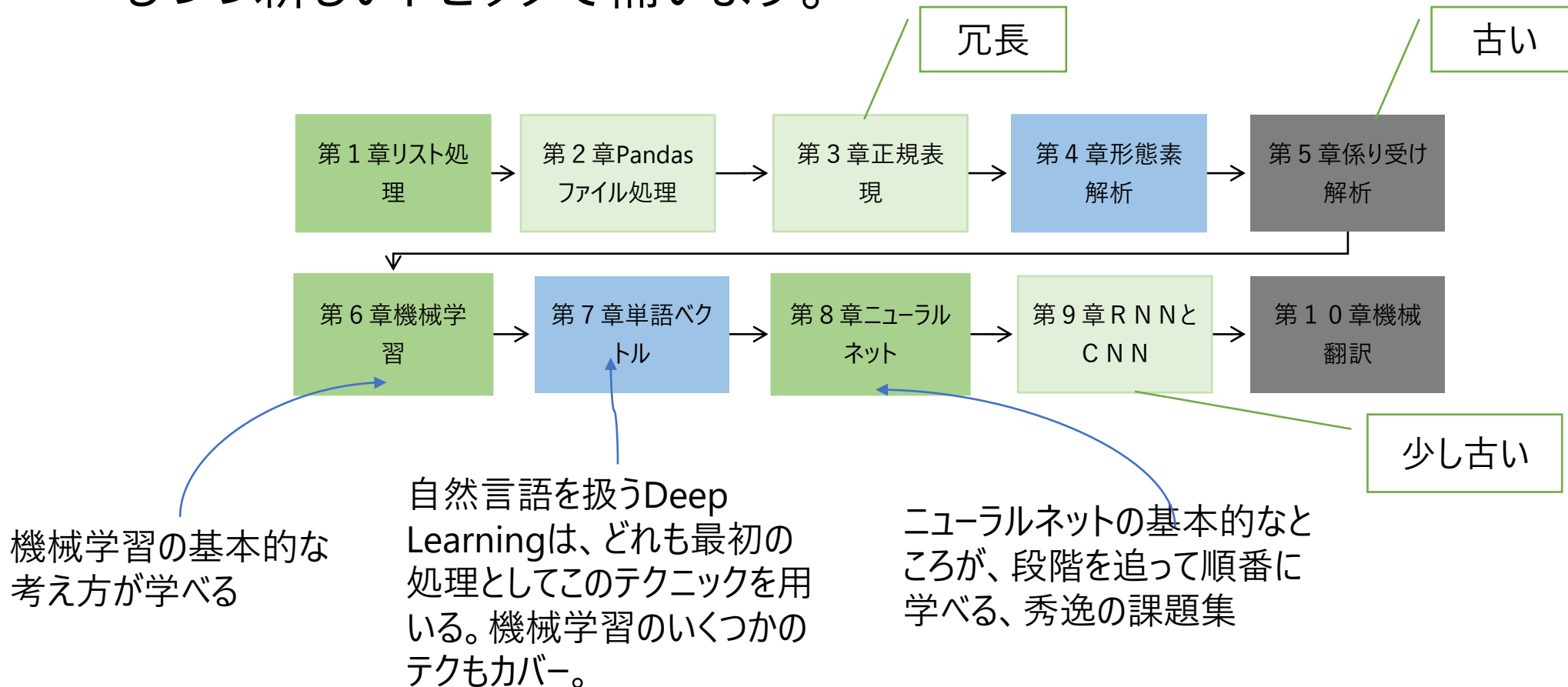
- 授業の教材として、主に、最近の技術を扱ったサンプルや課題を利用し、まず手を動かし経験することを重視します。ハンズオンというやつ。

教材について

- 前提知識は、必要に応じて、スライド・動画で解説します。
 - ネットに十分質の高い教材がフリーでのっている場合（Python解説など）、あるいは特定分野専門の先生の解説でいいのであれば（数学とか）、それをご紹介します。
 - 講師のほうがうまく説明できると思ったものは、講師の作った解説動画を紹介します。スライドの内容に言葉で説明を加えたものです。スライドをただ読むよりも、耳も使ったほうが記憶に残るので、スライドだけより動画を活用してください。
- 演習課題は、「言語処理 1 0 0 本ノック」
<https://nlp100.github.io/ja/> という課題集から選んだものを主に利用します。さらに、1 0 0 本ノックではネタ的にプアあるいは古いトピックは、PyTorchのチュートリアルサンプルなど他から補充します。

「言語処理 100本ノック」課題集

- いいところだけ活用し、冗長なところは適宜スキップし、古いところはスキップしつつ新しいトピックで補います。



注： 1 0 0 本ノックの課題集の章立てと学習 ロードマップとの関係

- 学習ロードマップに沿って学習を進めてもらい、その過程で適宜 1 0 0 本ノックの課題を演習として利用します。が、1 0 0 本ノックは課題の都合によって順序付けられています。以下のことにご注意ください。
 - 機械学習のクラスタリング、次元圧縮は、1 0 0 本ノックの課題集「単語ベクトル」の中で扱われていて、単語ベクトルをデータにして動かしてみるという位置づけになっています。
 - 1 0 0 本ノックの「単語ベクトル」は、すでにあるものを利用するという課題になっています。そのため、単語ベクトルの課題をやってからニューラルネットの課題をやるという順序になっています。しかし、本当の応用を組むには、応用に最適な自作単語ベクトルが必要です。それを作るにはニューラルネットを使います。そのため、学習の順序としては、ニューラルネットを先にやってから、単語ベクトルの課題に取り組んでください。

3つのコース

Pythonをマスターしたい人

- 準備Pythonのところを、じっくり時間をかけて取り組み、完璧に理解してください。

機械学習、ニューラルネットを使えるようになりたい人

- 準備Pythonのところを軽くやった後（必要に応じて戻る）、学習ロードマップの図の赤いフォントの部分を丁寧にやってください。進みが早い人は、ネットワークアーキテクチャを軽くかじるところまで。

自然言語アプリをかけるようになりたい人

- 青い部分が自然言語処理に特有の部分です。それらを含めて、順序だてて、学習してください。量が多いので、浅く広くカバーして後で特定トピックを深くでOKです。

演習の形式

- 演習は、準備フェーズのPythonのところは、1 0 0 本ノックを使ったゼロからの書き起こし。
- 基礎、ネットアーキテクチャのフェーズの演習は、サンプルコードの穴埋め。新しいことに慣れるには、ゼロから書くのは難しく、まずサンプルを読み慣れることから。

学習内容は全部オプション

- この授業は、高度なトピックまでカバーします。
- 一方、皆さんのバックグラウンド、習熟ペース、ゴールは、様々です。

学習は全部オプション

- 各人のバックグラウンド・習熟ペース・ゴールに合わせるため、事前知識の解説のどれを学習するか、課題（初歩的～高度）のどれに取り組むか、**学習内容は、全部、選択性**です。
- **ご自分のペース**で、選んだ解説を視聴・読んだり、選んだ課題に取り組んでください。まずはわからないことがあっても気にせず手早く全体像を把握しては、そのあとで、1個ずつ、特定のトピックに潜って腑に落ちるまでゆっくりじっくりやる、これを繰り返すやり方がいいです。

講師の役割は個別フォロー

- 行き詰った時の 1 on 1 のヘルプ
 - 誰かが引っ掛かりQ&Aを行った場合、ほかの人も同じところでひっかる可能性が高い。そのため、Q&A事項はクラスにメールかお知らせで共有していきます。
- 各人のゴールに沿った学習パスのアドバイス
- 巡回 1 on 1 面談
 - 人数が多い場合、これほとんどできないと思います。
- まとめレポートには全部目を通し、たいてい返事を書きます。また、必要なら次の回に 1 on 1 面談でフォローします。

講師の役割

- 皆さん、自分は理解力があるので、聞かなくても進める、聞くのも恥ずかしいしと、ひょっとして思うかも。そうなんです、皆さん、力があるので、自分で調べて一人で先に進めます。
- ですが、スライド・動画視聴中に理解できず、すっきりしない部分があって、調べてもほかにいい解説が見つからなかったり、また演習課題取り組み中に3分考えても指が動かなかった場合、講師をチャットで呼んでください。1on1のブレイクアウトルームに移ります。
- 皆さんの時間の節約と、講師の教材準備不足の部分の解消のためです。

教えあう仕組み？

- 講師との 1 on 1 でなく、学友に聞いたり教えあったほうがやりやすいかも。友達がいて、詰まったら、その友達と教えあってみてください。

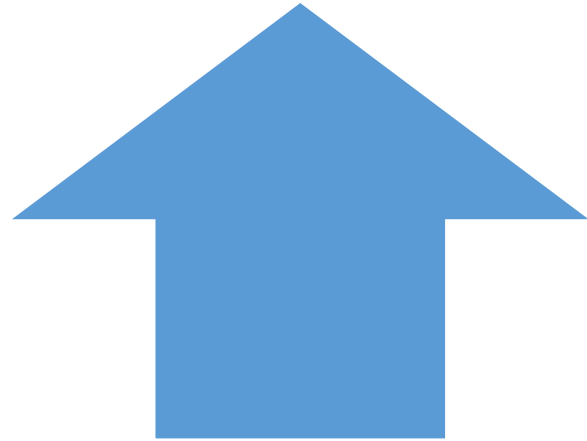
講師の状態

- 講師は、以下のいずれかの状態にいます。状態表示をしますので誰かのフォロー中以外は随時声をかけてください。
 - だれかのフォロー中
 - クラスメンバーの状況把握作業中
 - 待機中

モンテッソーリ教育

- 自発的な成長意欲
- 成長段階に応じた教具を提供

経験上の効果

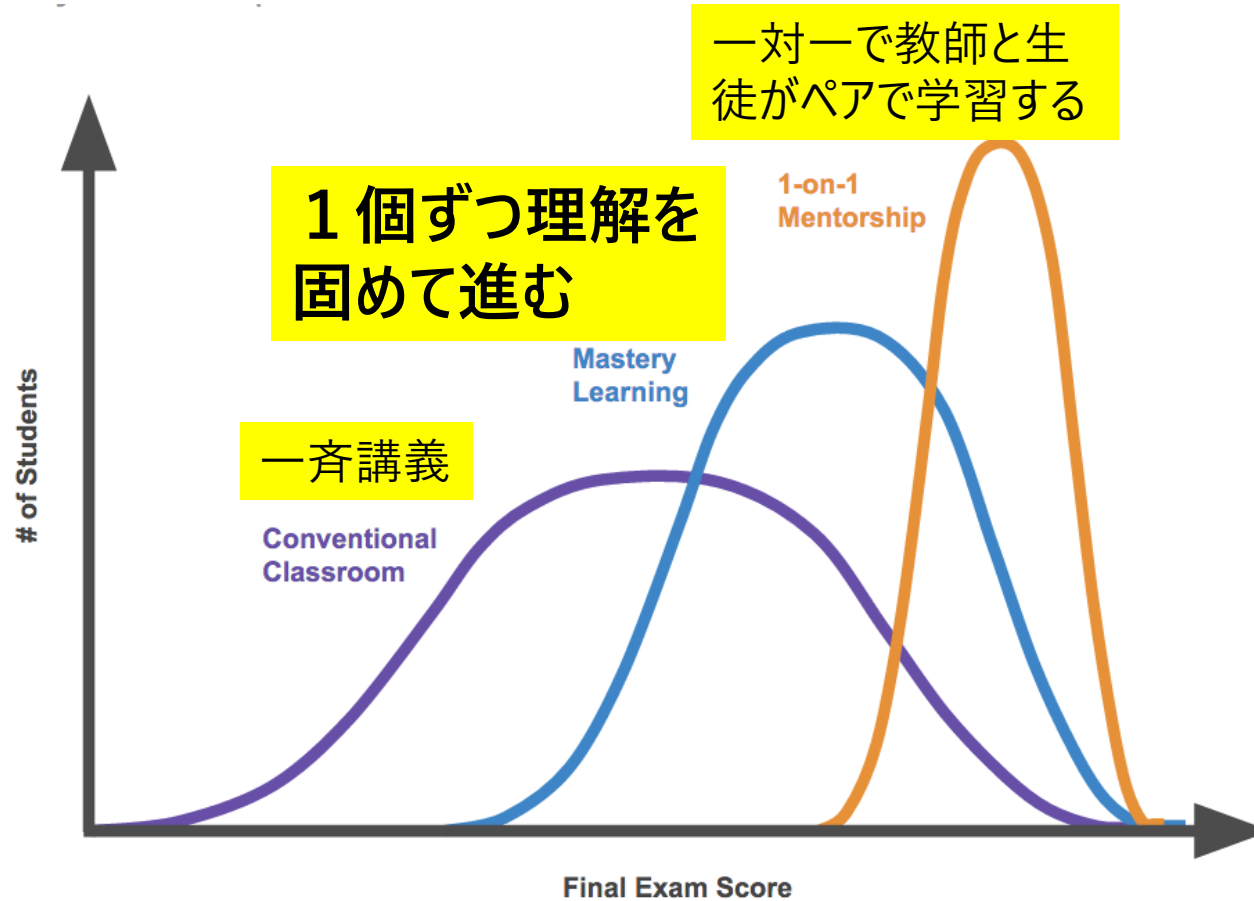


自分の興味に基づいて、自分で目標を立てて、
自分のペースで学習する、このやり方のほうが、
ほとんどの人が高度なトピックまで消化できます。



経験上、一斉授業だと、高度なトピックまでつ
いてこれるのは、1 割程度です。

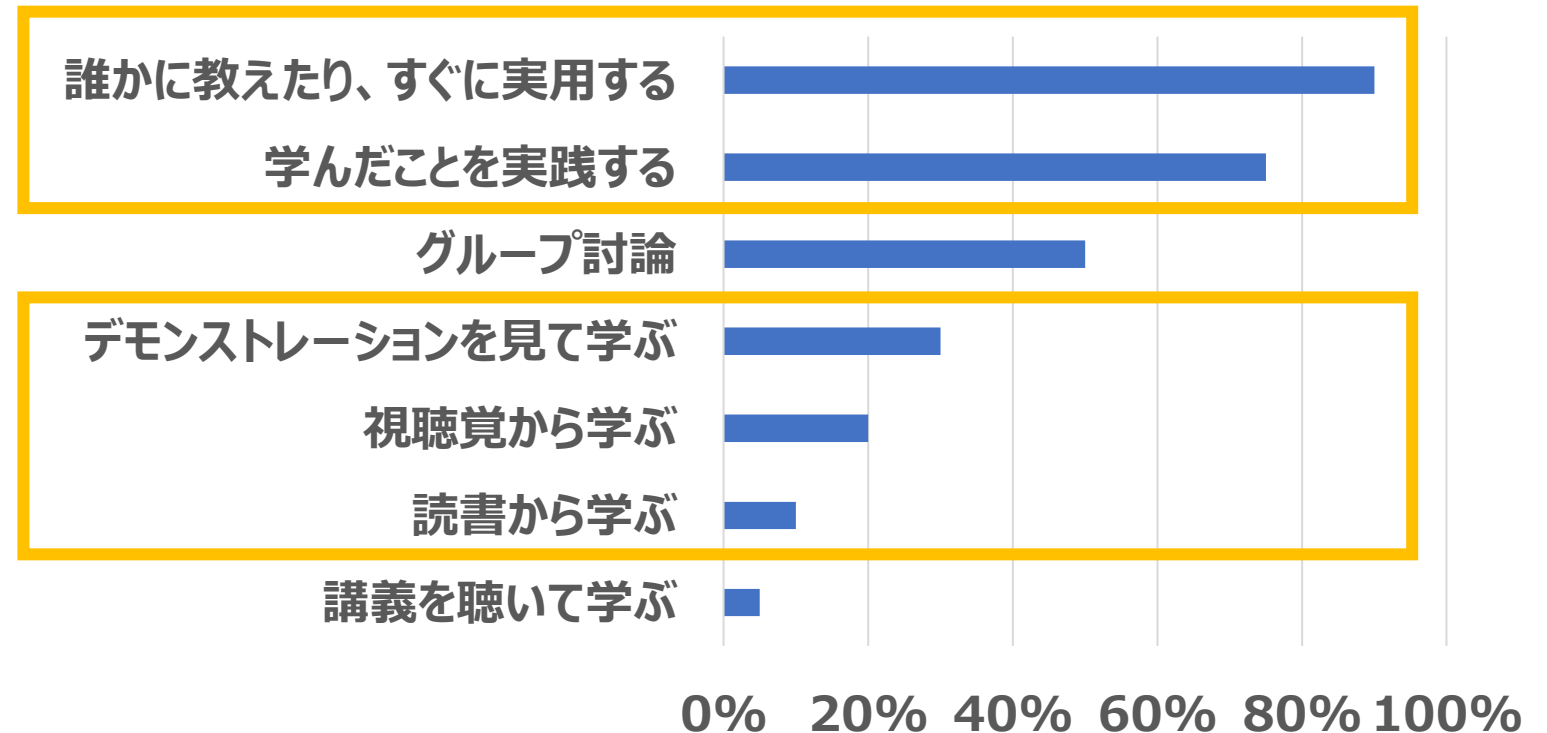
各人が各人のペースで、1個ずつ理解を進める やり方が効果ある



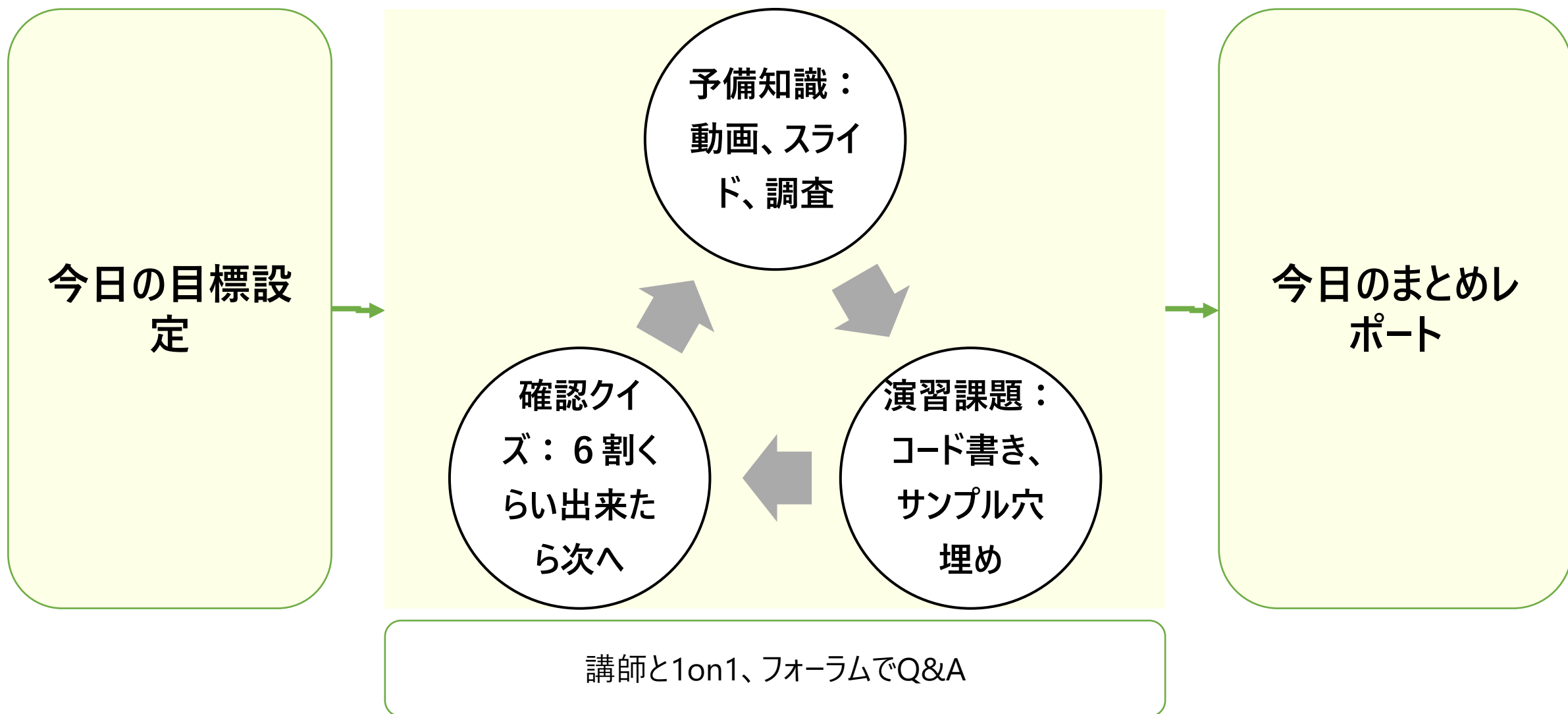
この授業で知識を得たら、自分の課題に取り組める状態になっています。この授業はそこへの道案内。

卒研や社会に出て、自分の課題に取り組むときに、一番力がつく

この授業



授業の流れ



単位認定？

- 取り組むトピック、演習課題、到達目標のどれも選択制です。そのため、単位認定で一律の試験問題はやりません。
- また、提出課題**数**やクイズ**正答率**も、各人の学習スタイルの選択に依存するので、成績評価には使いません。

単位認定？

- 主に、以下の評価の和でABCD判定します。
 - **最後**に、各人の初期値に比べて、どのくらいこの授業を役立ててもらえたかを、文章で**レポート**してもらいます。15点。
 - また、**毎授業の設定目標とまとめレポートの内容**によって、2 時間を有効に使ってくれたかどうかを判定します
 - 計画提出1点、まとめ提出2点
 - 初回はまとめのみで3点 + 3点 x 14回で配分45点。
- 出席率を加味します。40点。

学びの手引き

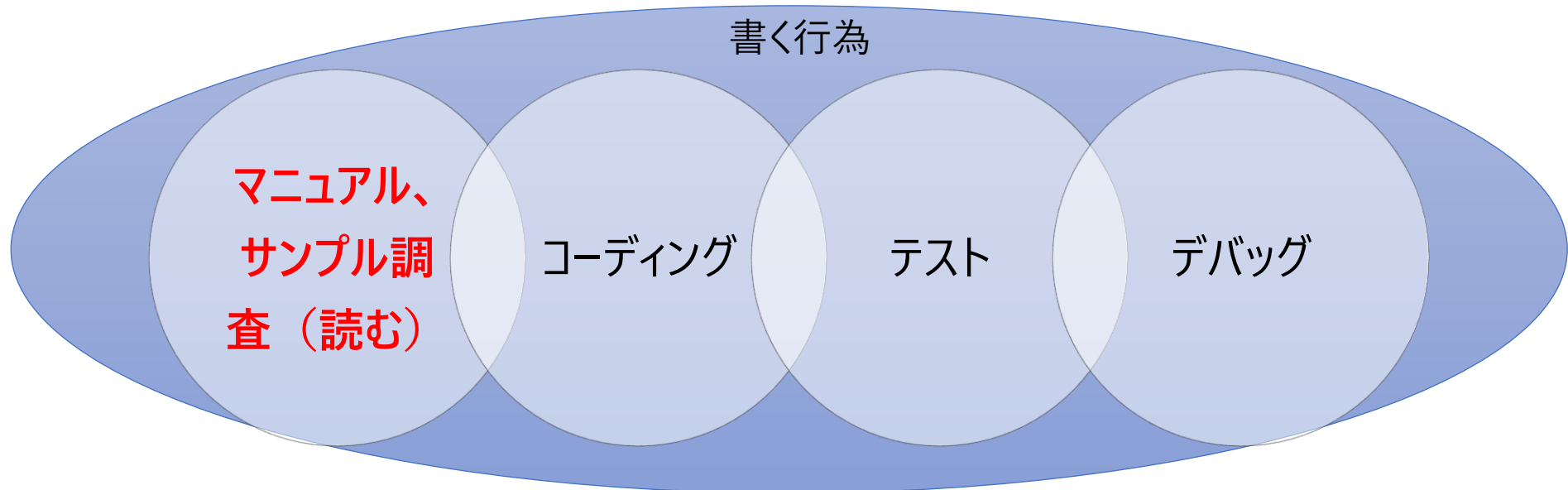
導入ヒヤリング 2

Pythonの読み書きに自信がありますか？

この書き方、習ってないから知らないもん？

プログラミング言語の習得

- プログラムを書くとき、たいてい、書き方がわからないケースに遭遇します。新しい知識を習得するには、以下の二つの調査を行い、慣れていくしかありません。
 - オンラインマニュアルで調べる。
 - 検索して、先輩の書いた解説やサンプルコードを調べる。まず見つかります。



プログラミング言語の習得

知らないとネガティブにとらえず、知らないことを一つ見つけて（無知の知）、一つ新しいことを学ぶ機会を得たとポジティブにとらえてください。

プログラミング言語の習得

調べたときに、たとえきっちり理解できなくても、使い込んでいけば、赤ちゃんが言葉を覚えるのと同じで、自然と身についてくるので、直ちに悲観する必要は全くありません。

print文やデバッガのステップ追跡で、動的に結果を確認すると、理解できることがあります。

新しいものに慣れるやり方

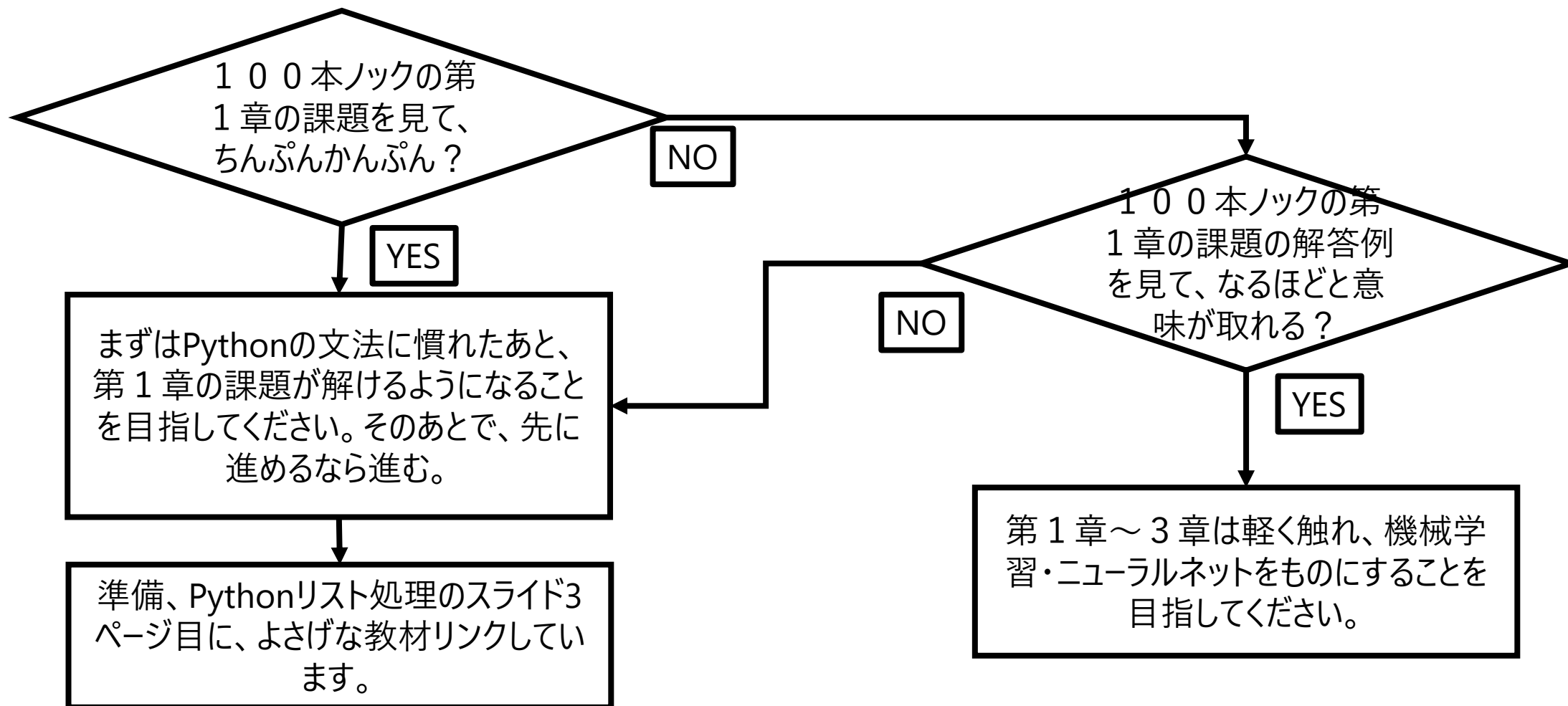
大枠(Python)に慣れていて、その中の特殊なパッケージ(numpy、OpenCVとか) などの場合

- まず使ってみる。必要に応じて、調べて、理解する。そうやって、具体事例の理解を積み上げて、理解を広げていく。

枠組み自体が新しい場合：初めてAndroid プログラミングをやるとか、初めてPyTorchでDLを書くとか

- ある程度、ドキュメントで全体の概要・骨格を把握してから、サンプルを手掛かりにして、個々の問題に取り組む。

Pythonに慣れることを目標にするか？



導入ヒヤリング 3

数式が苦手？

数学の前提？

ネットワーク構成を理解するためには、各パーツの動作や意味も理解していると、全体の動きがより理解しやすくなります。

コードの中身にきちんと潜るには、線形代数と微分の知識が必要になります。しかし、この授業では、ネットワークの構成を概略把握することに重点を置き、数学の話は必要最低限だけに絞ります。

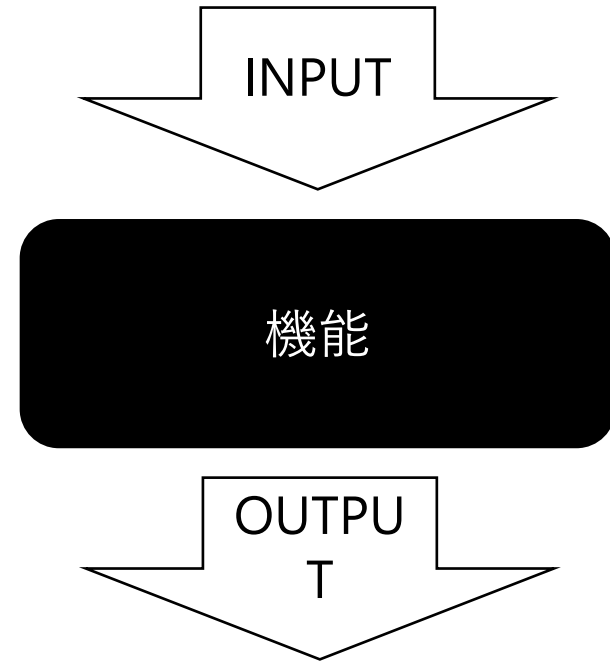
数学の前提？

- 意欲のある人は、高校数学の学びなおしのために、無料の <https://edupa.org/> の『高校数学標準講義』をお勧めします。その対数・指数、ベクトル・行列、微分のところ。
- また、大学の教養課程の数学は、YouTubeなどで探せば講義の記録が見つかります。偏微分の考え方だけ。

数式が苦手？

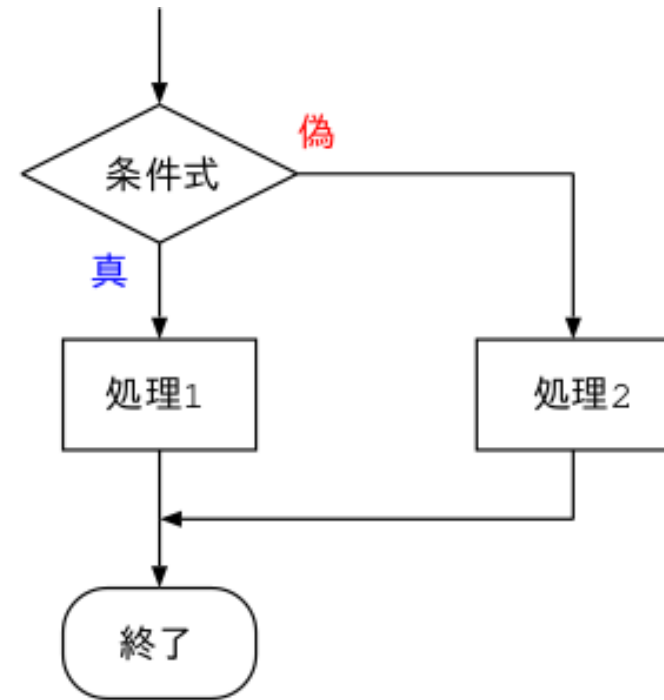
数式は、メソッド。まず機能（何をするためののか）、入出力はなにで、入出力はどう対応するか、というメソッドの仕様を把握し、腑に落ちたら、中身を知らなくても、もう使えます。

例：Softmaxの機能は、任意の個数の数値を、合計で1となる数値に変換すること。それによって確率値と解釈できる量にする。



数式が苦手？

- 式の中の関数の組み合わせは、いわばメソッドの実装。中の処理フローを理解するのは、アルゴリズムの理解と同じ。具体的なテストデータを使って、処理ステップをトレースすると、理解できる。



数式が苦手？

- 余裕があれば、いろいろな設計の中でなぜその式なのかを調べてみるとよい。
 - 例：softmaxで、なぜネイピア数あるいはオイラー数 e の冪を使うのかは、学習（逆伝播）のための微分結果が扱いやすいから。

強化学習？

- 今どきの強化学習は、深層学習ニューラルネットを関数近似として実装に利用しています。
- 強化学習に取り組むには、まずは第 8 章あたりまで把握してからにしてください。

環境

言語：Python3

ランタイム：Google Colaboratory

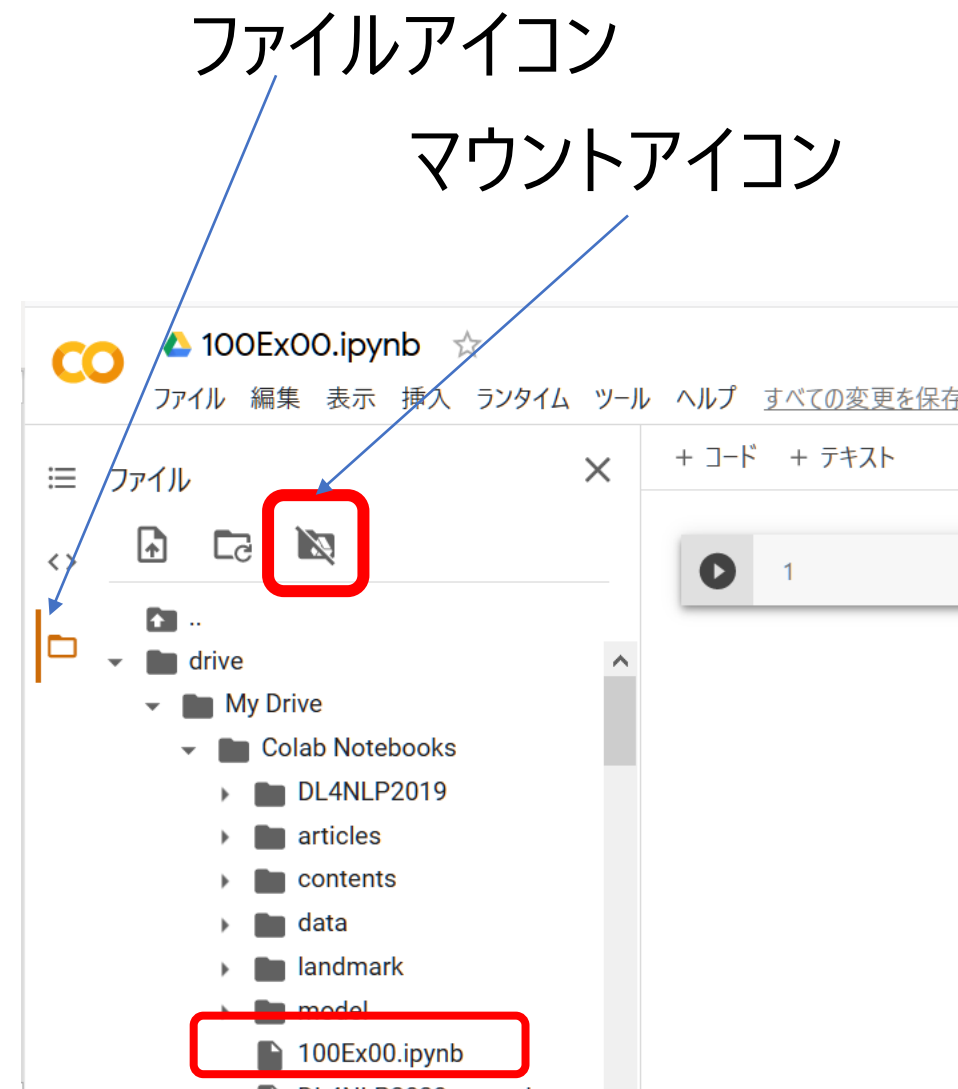
Google Colaboratory

導入：作業 1

- ブラウザで Googleのアカウントに入ります。
- Google Colaboratoryを検索し、
<https://colab.research.google.com/notebooks/intro.ipynb?hl=ja> を開いてください。
- 最初のコードブロックの上にカーソルを持っていき、三角アイコンをクリックして、
`seconds_in_a_day = 24 * 60 * 60` `seconds_in_a_day` を実行してみましょう。
- ファイルメニューから「新規ノートブックを作成」を選び、左上のファイル名
`Untitled0.ipynb` を `sample.ipynb` にしてください。
- 各ノートブックには、Python込みのランタイム環境が張り付きます。メモリ上の処理だけなら、このままで動きます。一方、ファイルIOを行う際、Google Driveをマウントする必要があります。カレントディレクトリは、`/content/drive/My Drive/Colab Notebooks` で、このパスを意識します。
- また、ランタイムはクラウド上の資源で動いているので、ローカルのGdriveにデータを置いて、関連ファイルがクラウドと同期済みかどうかを意識します。

Gdriveのマウント

- 左のファイルアイコンをクリック。
- 左に展開したタブの上部にあるマウントアイコンをクリック。
- 自分のdrive>MyDrive> Colab Notebooksフォルダー下にsample.jpynbが置かれていることがわかります。
- -----
- これから授業で提供したノートブックは、ここにコピーを保存して、課題を書き込んでから、スタログへアップロードして提出となります。



導入：作業 2

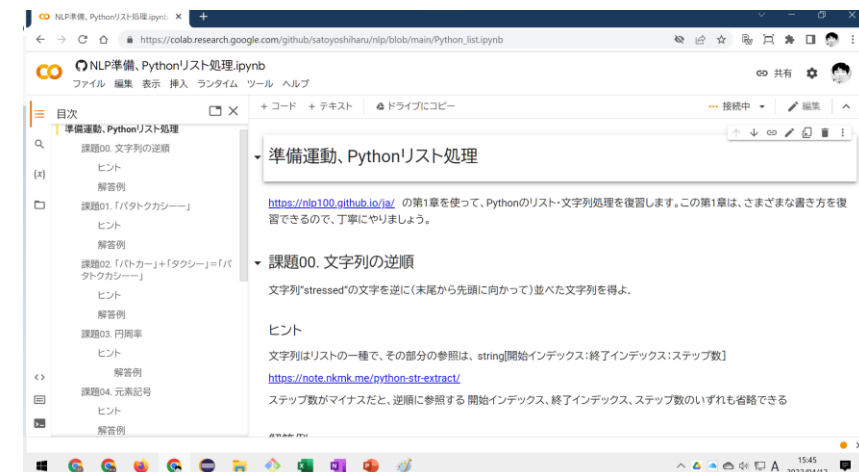
- Google Collaboratoryの操作に慣れます。
 - 目次表示
 - ファイル表示
 - 「+コード」で、コード挿入。コードブロック上にカーソル持って行って右向き三角のクリックで、コード実行。
 - 「+テキスト」で、テキストブロック挿入。簡単な成形（マークダウン）構文が使える。
 - ランタイムのGPUへの変更

課題提出

- スタログ、「提出、確認クイズ」フォルダーの提出箇所へ提出する
- 100本ノックの課題は、解答（もしくは穴埋めた）コードを入れた、Jupyterノートブック (.ipynb) のファイル提出。
- 確認クイズは、コードの場合と選択式クイズの2種類あります。
 - コードの場合は、ipynbのファイル提出。
 - 選択式クイズは、スタログのクイズとして登録しています。GitHubには.txtでクイズ原稿を載せていますが、それではなくスタログのほうへ直接解答してください。

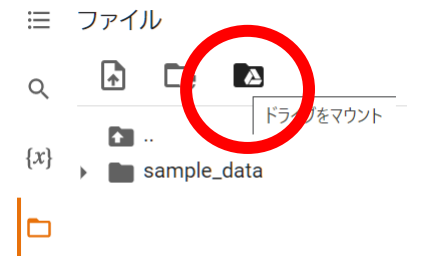
ノートブックの提出

1. <https://satoyoshiharu.github.io/nlp/> へアクセスする。
2. 自分の取り組むトピックに応じ、「Pythonリスト処理課題ノート」などをクリックします。
3. すると、Google Collaboratory 経由でノートブックが開きます。
 - Google CollaboratoryとGitHubは統合していて、URLに細工することで、CollaboratoryにGitHubのファイルのパスを引き渡しています。



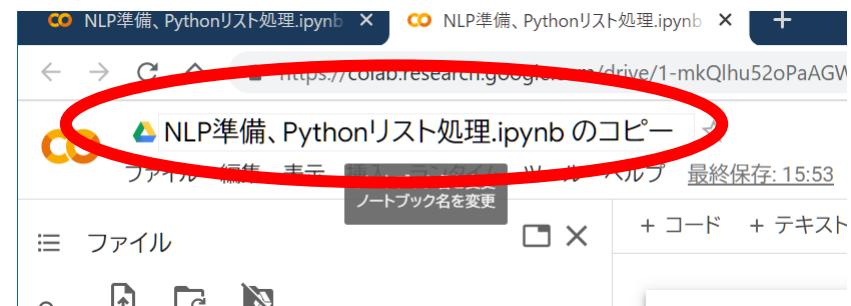
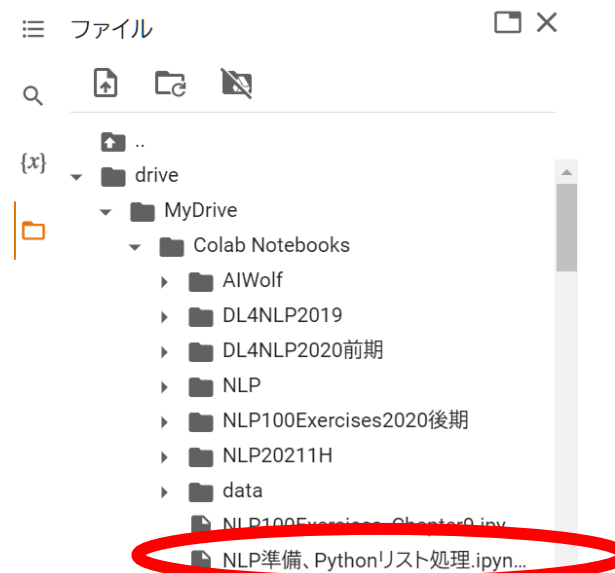
ノートブックの提出

4. ここで、そのままソースを上書きすることを心配せず編集・変更・実行できます。が、提出のため、最初に、ローカルファイルを自分用に作ってしまいます。「ファイル」メニューから「ドライブにコピーを保存」を選択してください。
5. 別タブにコピーファイルが開きます。これを自分のドライブに保存します。
6. 左端のファイルアイコンをクリック、開いた左ペインの上のほうにあるドライブのマウントアイコンをクリック。
7. このコード実行せよときたら、三角をクリックで実行。Google Driveへのアクセスを許可かというダイアログが出たら、許可。



ノートブックの提出

8. 1, 2 秒すると、左のフォルダー改装に、「drive」が登場します。それを展開して、**drive/MyDrive/Colab Notebooks/** の下に、先ほど作ったローカルなコピーを見つけます。「Pythonリスト処理課題ノート」でしたら、「NLP準備、Pythonリスト処理.ipynbのコピー」とかいうやつです。
9. 左上のファイル名表示のところで、「Pythonリスト処理.ipynb」などと適当な名前に変更します。以降、Pythonリスト処理.ipynb を編集しては、「ファイル」メニューの「保存」。



ノートブックの提出

10. ファイル編集が終了したら、

- もしもGoogle Driveをローカルなディスクと同期設定している場合は、そのローカルなフォルダーのファイルを直接、スタログにアップロードします。その時ファイルのタイムスタンプを見て、クラウドと同期済みかどうか確認してください。
- そうでない場合、「ファイル」メニューの「ダウンロード」で適応なところにいったん置いてから、それをスタログにアップロードします。

導入まとめレポート

- 以下を文章にして提出してください。
 - 自分がこの授業でやりたいこと
 - 自分の思ったことや意見
 - 理解できなかったことや残った疑問