情報科学概論　第２回

情報システム

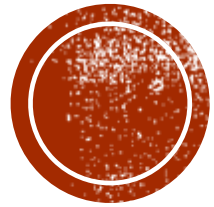
+ 人工知能システム

立教大学大学院 人工知能科学研究科

２０２２年４月２４日

アンドラーデ ダニエル

andrade@rikkyo.ac.jp



自己紹介

ドイツのバンベルク
周辺で生まれた。

- 名字：**Andrade** Silva
  アンドラーデ　シルバ
- 名前：**Daniel** Georg
  ダニエル　ゲオルグ
- 出身：ドイツ
- プライベート：子育てでへとへと
  （男4歳、女1歳半）

- 2007年 ドイツのパッサウ大学で情報科学（Diplom）修了後来日
- 2011年 東京大学で博士（情報理工学）専門：自然言語処理
- 2011年から2021年まで、NECの中央研究所で研究員
- 2019年　総研大（統数研）で博士（統計科学）専門：モデル選択
- 現在は　広島大学の准教授、立教大学の客員准教授
  - 「情報科学概論」with 大西先生
  - 「人工知能科学特別演習」（← 不確実性の定量化、モデル選択など）

研究内容

- 過去にはテキストデータにおける分類、翻訳、意味理解に関する研究。
- 近年はベイズ統計に基づいたモデル選択、AutoML（機械学習を実問題に適応する簡略化方法）、自然言語処理＋数字データなどに取り組んでいる。

アンドラーデの担当授業に関する参考文献

- 今日：

  第２ 章「情報システム」

  山口和紀, 情報第2版,　東京大学出版会(2017)

- ５月１日：

  第４ 章「情報の伝達と通信」

  山口和紀, 情報第2版,　東京大学出版会(2017)

- ５月８日：

  第５章「計算の方法」

  ＋　第６章「計算の理論」

  山口和紀, 情報第2版,　東京大学出版会(2017)

本日の内容

1. 情報システムの定義と構造
   - 集中型と分散型
   - クラウドコンピューティング
2. 情報システムの開発
   - 情報システムの開発プロセス
   - 人工知能システムの開発プロセス
3. 情報システムにおける計算方法
   - ビッグデータ
   - 並列と分散処理
4. 人工知能システムと学習データの重要性
5. 情報・人工知能システムにおける安心・安全性

参照文献：
第２ 章「情報システム」
山口和紀, 情報第2版, 東京大学出版会(2017)
第４ 章 「通常のシステムと人工知能システムの開発プロセスの違い」
本橋 洋介, 人工知能システムのプロジェクトがわかる本 企画・開発から運用・保守まで, 翔泳社
“Understanding artificial intelligence ethics and safety - A guide for the responsible design and implementation of AI systems in the public sector”, The Alan Turing Institute
https://www.turing.ac.uk/sites/default/files/2019-06/understanding_artificial_intelligence_ethics_and_safety.pdf

学習目標

- 集中型と分散型のメリットとデメリットを意識した情報システムを設計できる。
- 従来ビジネスモデルと比べて、クラウドコンピューティングの経済的・技術的なメリットを説明できる。
- なぜ大規模のデータ処理に並列・分散処理が不可欠かを説明できる。
- 並列アルゴリズムの計算コストを求められる。
- 従来の情報システムと人工知能システムにおける開発プロセスの違いと共通点を説明できる。
- 情報システム運用上ではどのような危険性があるかを説明できる。

本日の授業運用

- 授業時間：５時限（17:10〜18:50）
- 授業中、ChatやSlackが見えない可能性がありますので、
  キリのいいところで質問にまとめて答えます。
  スライド上では「質問タイム」で表示します。
- 「質問タイム」では是非音声でも質問をお願いします。
- Breakout Roomを一回5~10分間に開催しようと思います。
- 重要な専門用語の表示はできるだけ英語または日本語に英語をつけることにしました。
  - 英語で論文を読む際には役に立つ。
  - まだ正式な和訳がなく、対応している日本語の言葉には複数あったりする。
  - とはいえ用語を暗記するより、概念を理解することが重要。
- ぜひ今後の改善のためにSlackや私のメールアドレス（andrade@rikkyo.ac.jp)に意見・要望などを送っていただければ改善に努めます。

1.情報システムの定義と構造

ICT（INFORMATION AND COMMUNICATIONS TECHNOLOGY）

定義：

- 情報通信技術。電子機器間の通信技術の全般とそのインフラ（インターネット,G5など）を表す。
- IT (Information Technology) とほぼ同様の意味。（ICTでは通信の概念が強調されているだけ）

情報システム (INFORMATION SYSTEM)

- 定義（＊）：
  「**コンピュータを中心**とする情報を処理する機器と、情報を伝達する**ネットワーク**を組み合わせて、様々な**サービス**や機能を提供するシステム。」
- ここ３０年「**ネットワーク**」が「**インターネット**」と等しくますます日常に浸透。
- 例：
  - 有料道路料金自動徴収のETC （自動車が料金所に無線通信で通過を要求する等）
  - ゲーム機（インターネットを通して友達とゲームをする。）
  - ポケモンGO（位置情報を測定するGPSと, Google Mapsと, 課金システムが連携しているシステム）
  - クレジットカードの決済（店舗からクレジットカード会社に支払い要求が送信され、その決済依頼が自動的にカード会社で処理され、その結果が店舗に返信される。）

※ 人工知能システムは情報システムの中に含まれている。

（＊）「情報」ページ１２枚

集中型と分散型の情報システム

(**CENTRALIZED** AND **DISTRIBUTED** INFORMATION SYSTEM)

- 情報システムを実現するために利用されている形態は概ね以下に分類できる。
  - 集中型：主に一箇所（サーバなど）で処理を集中させる。 →P11
  - 分散型：複数の平等の立場を持つ機械で処理を行う。 →P12

集中型 (CENTRALIZED SYSTEM)

- 端末(Terminal）からホスト(Host)にアクセスして、処理を行う。
- 重たる計算やデータの保存はホストで行われ、端末上の必要な計算が少ない。

- 例：
  - 端末：手元にあるラップトップ　　ホスト：スーパーコンピュータ。
  - Client/Server （次のスライド）

- メリット：
  - 端末の生産コストが低い。
  - データの整合性が取りやすい。

- デメリット：
  - 計算処理能力が拡張しにくい。（スーパーコンピュータAをスーパーコンピュータBと置き換える）
  - システムの反応がネットワーク上の通信速度に左右される。

分散型 (DISTRIBUTED SYSTEM)

- データや処理が複数のノード（PC・サーバなど）に分散されている。
- ノードがネットワーク上で繋がってデータの処理や保存を分担している。

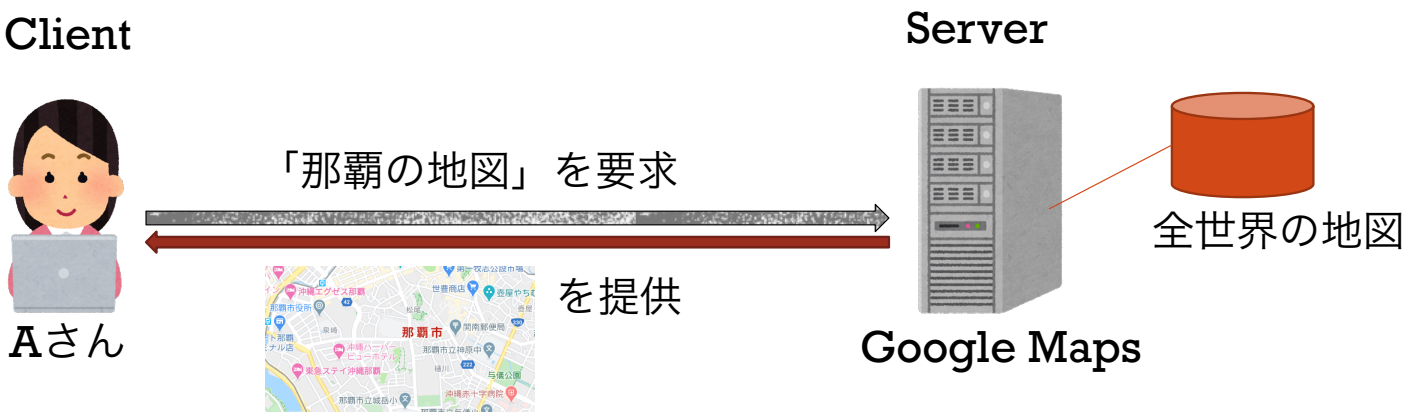
- 例：
  - インターネット（Web pageが複数のサーバに保存され、複数の経路で繋いでいる。）

- メリット：
  - 計算処理能力が拡張しやすい。　（パソコンA, B, Cにさらにパソコンを加える）

- デメリット：
  - 整合性を取るために、規則やノード間の通信が必要。(Protocolや規格)
  - 処理能力がネットワーク上の通信速度に左右される。

集中型と分散型の組み合わせ（1/2）

- 多くの現在のシステムは、見方によって、集中型とも分散型とも言える。
- 例：Client/Serverのモデルは場合によって集中型でもあり、分散型でもある。

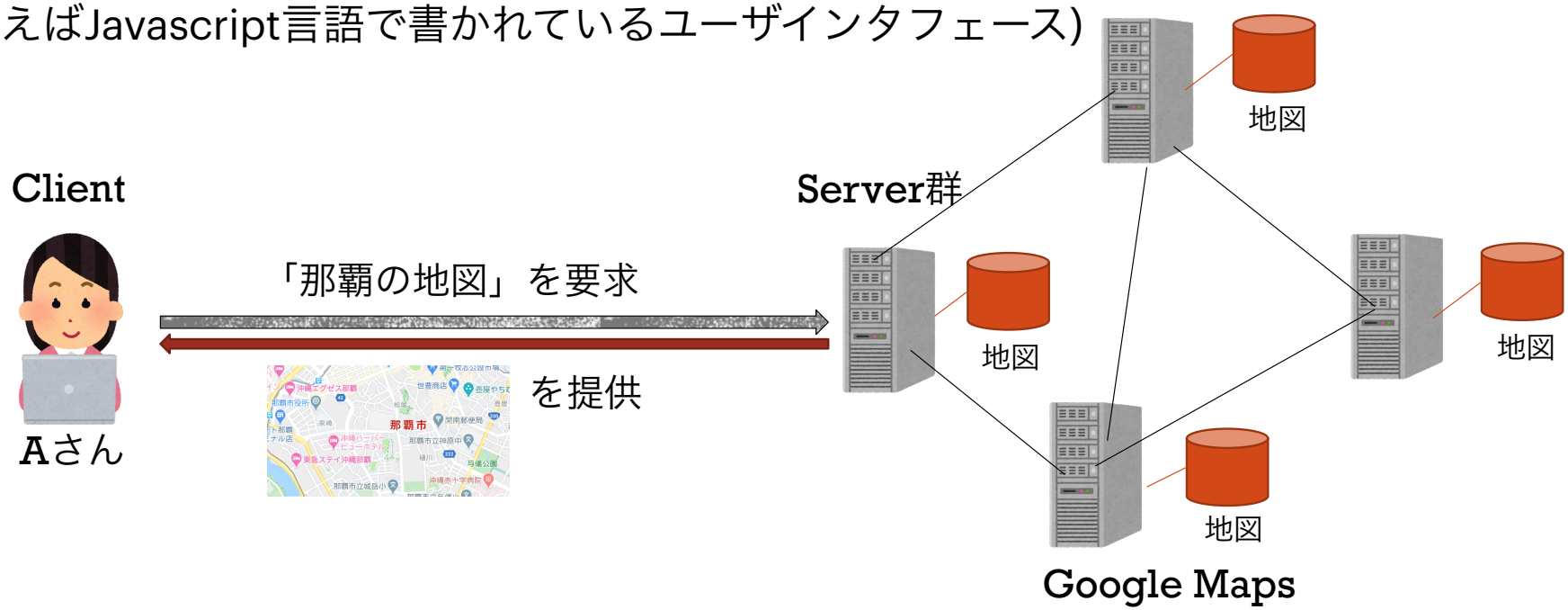


- 集中型のように見える：
  すべての処理がサーバで行われて、クライアントに送信される。

集中型と分散型の組み合わせ（2/2）

- 実際には分散型でもある：
  - 地図の索引や画像データが複数のサーバに分散されている。
  - ウェブブラウザ上でも計算が行われている。

    (例えばJavascript言語で書かれているユーザインタフェース)



クラウドコンピューティング

(CLOUD COMPUTING)

- **ソフトウェア**や**データの保存**がローカル（手元のパソコン）の代わりに、**インターネットを通して**提供される。
- 例：
  - ソフトウェアをインストールせずに、Google Docsではワープロの機能が提供されている。
  - ローカルなHDD(ハードディスク)の代わりに、DropboxやiCloudにデータを保存する。

- Cloud Computingの一種：
  - SaaS（Software as s Service)
  - DaaS (Data as a Service)
  - IaaS (Infrastructure as a Service)
  - PaaS （Platform as a Service)

SAAS (SOFTWARE AS A SERVICE)

- 顧客や人事管理ソフトウェアといった企業向けのソフトウェアがクラウドのサービスとして提供される。
- 一回払いではなく、企業が必要な機能・キャパシティに応じてサービス料金を払う。
- 経済的なメリット：
  - ソフトウェアの実行に必要なリソース（プロセサやHDD）の購入が不要になるから、必要な時だけ必要な分を払う。
  - ビジネス拡大に随時対応できる。

DAAS (DATA AS A SERVICE)

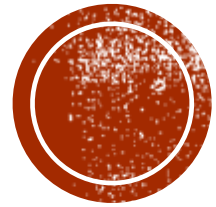
- 主に企業向けにデータが提供される。
- SaaSと同様に必要な分だけの支払いが必要。
- 例：地図データの提供サービス。

IAAS (INFRASTRUCTURE AS A SERVICE)
PAAS (PLATFORM AS A SERVICE)

- コンピュータ資源を提供してくれるサービス
- SaaSと同様に必要な分だけの支払いが必要。
- 例：AWS (Amazon Web Services),

  Microsoft Azure,
  Google Cloud

質問タイム



BREAKOUT-ROOM 演習問題

- 携帯電話で自分の撮った写真に友達が写っているかどうかの判定システムを開発。

  A) Client-Serverのシステムではデータ処理がどこで行われているか。データの通信速度はどれぐらい必要でしょうかか。

  B) すべての処理が携帯電話で行われている場合ではどのようなメリット・デメリットがあるか。

  A) to B) のそれぞれの短所と長所を列挙してください。

- Client-Serverのシステム：
  撮影した写真をクラウドに保存した写真と照合する。必要になる分類・アラインメントはすべてクラウド上で計算される。

- 携帯電話内に限定したシステム：
  撮影した写真を携帯内に保存した写真と照合して、一致すれば、
  ユーザに「友達ですか」と聞く。

議論

- AIにおける企業を開設にあたって、以下の二つの選択に関する利点と欠点とは？
  - 開発用にサーバ(GPUマシンなど）を購入する。
  - サーバの購入の代わりに、AWSといったクラウドのサービスプロバイダを利用する。

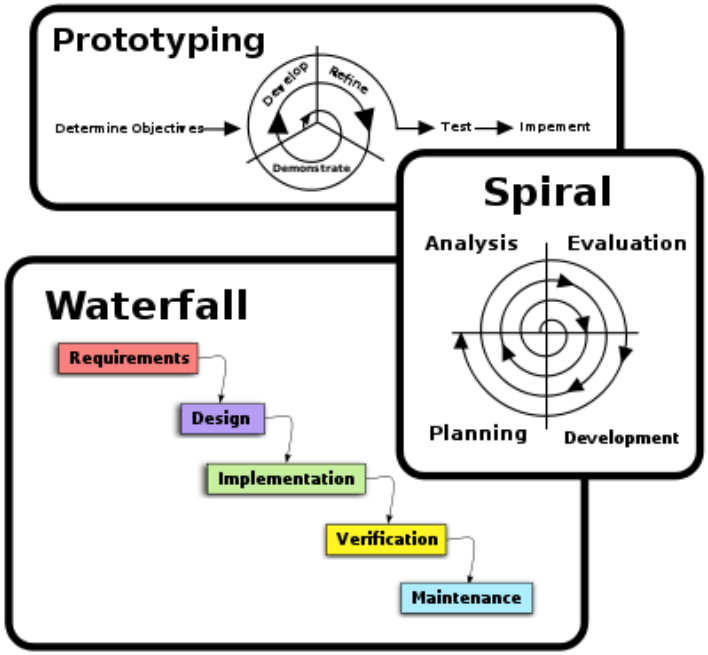
- クラウドサービスの欠点：
  - データ（ソースコードや顧客データなど）が全てクラウドに保存されるので、データの安全性（サイバ攻撃への対策）とアクセス性に関してサービス・プロバイダに信頼できるかを精査する必要がある。
  - データへのアクセス速度がインターネットとの接続速度に左右される。
  - ローカル（＝手元）のサーバほど環境の設定などが自由にできないため、利便性に欠けるかもしれない。

- クラウドサービスの利点：
  - サーバへの高額な投資を避けられる。
  - サーバの保守にかかるコストを省く。
  - サーバの更新コストが不要。
  - 計算能力の増減に伴って、契約しているリソースを柔軟に変更できる。

2. 情報システムの開発

情報システムの開発プロセス（SOFTWARE ENGINEERING)

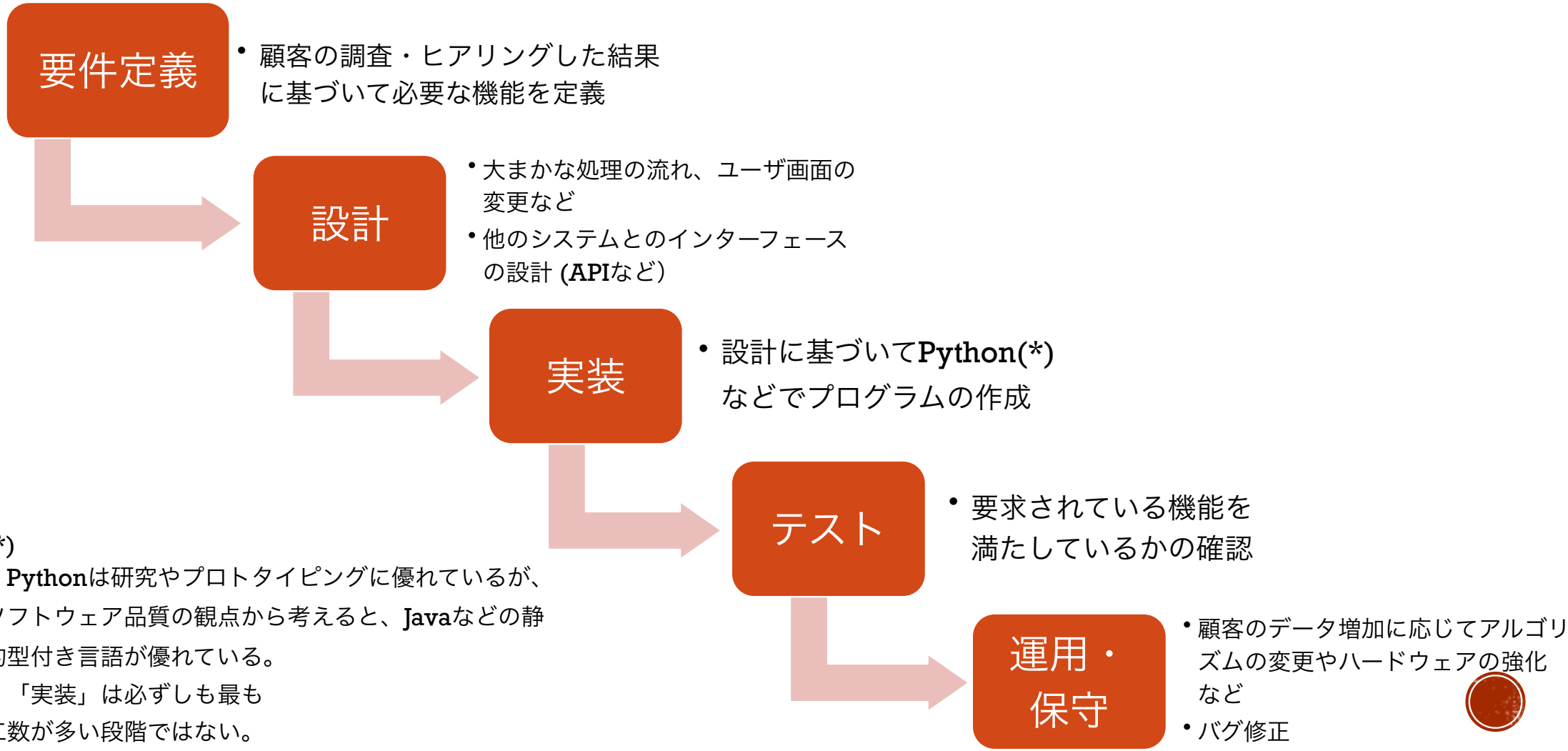
- 大きな情報システムの開発を効率化するために、様々なプロセスが提案されている。最も典型的・代表的な工程はウォーターフォール型プロセス (Waterfall Model)。（p21）



https://ja.wikipedia.org/wiki/ソフトウェア開発方法論 より

- 多くの工程で共通している段階は「要件定義」，「設計」，「実装」、「テスト」、「運用・保守」。

ウォーターフォール型の開発プロセス



(*)
・Pythonは研究やプロトタイピングに優れているが、ソフトウェア品質の観点から考えると、Javaなどの静的型付き言語が優れている。
・「実装」は必ずしも最も工数が多い段階ではない。

AGILE SOFTWARE DEVELOPMENT (アジャイルソフトウェア開発)

- 設計の間違いなどにより柔軟に対応できる開発方法
- ウォーターフォール型と違って、各コンポーネントの実装と共にテストが行われている。
- 早く問題を発見して、早く対応する。
- 実装した**暫定的**なソフトの**予備**実験・検証をもとに設計を変えたりする。

Fail Fast
Fail Cheap
Fail Smart
のマントラに沿った開発方法

人工知能システムの開発プロセス (1/2)

従来の情報システムの工程に入る前に「企画」と「トライアル」がある(*)

1. **「企画」**
   - 人工知能の役割を明確にする。役割には大きく分けて２つある。
     - 人間を支援するシステム（例：商品の推薦システム）
     - 完全に自動的に動くシステム　（例：自動運転）
   - 人工知能（=機械学習）に必要な学習データをどのように取得できるか、必要な量を十分に確保できそうかを検討。
   - 人工知能の性能目標などを立てる。
2. **「トライアル」**
   - 簡単なシステムや限定的な学習データを利用し、人工知能の有効性を確かめる。
   - 目標は達成できそうかを確かめる。
   - 予備実験の結果によって機械学習の具体的な手法(深層学習(NN)、SVMなど）を決める。

(*) 「人工知能システムのプロジェクトがわかる本」の第２章により

人工知能システムの開発プロセス (2/2)

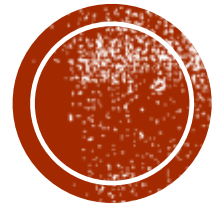
従来の情報システムの工程における「要件定義」，「設計」，「実装」、「テスト」、「運用・保守」は人工知能システムの開発にも適応されるが、「運用・保守」の段階では様々な注意点がある。 (*)

「運用・保守」の注意点

- 性能の劣化を監視。
- システムへの入力データが大きく過去のデータから異なった場合、学習データの更新が不可欠。
- NNのような学習モデルでは、性能に対する保証はなく、データの追加で、過去には正しく判別された事例が間違って判別されてしまう場合がある。
  特にNNのようなBlackboxモデルではエラー分析が困難。場合によって対処策がない。

(*) 「人工知能システムのプロジェクトがわかる本」の第２章により

質問タイム



3. 情報システムにおける計算方法

背景：ビッグデータ (BIG DATA)

https://www.cs.cmu.edu/~epxing/Class/10708-17/notes-17/10708-scribe-lecture22.pdf より（2017年以前の数字なので、もうすでに更新されているだろう。）

➢ **SNS**のデータ例:

➢ 社会のデータ:
- ➢ 店舗の購入履歴
- ➢ 医療のデータ
- ➢ 携帯・**Apple Watch**等から取得した行動履歴

➢ センサーのデータ例:
- ➢ **IoT (Internet of Things)**、**Smart Home**

➢ 科学のデータ例:
- ➢ 天文データ、遺伝子発現データ

COVID-19によって進むデジタル化の影響で**Big Data**の増加がさらに加速するだろう。

背景：ビッグデータにおける問題

- **処理速度：**
  単独のCPU・サーバでは処理仕切れない。
  対策として、 並列・分散処理。

- プライバシー：
  例えば、個人情報の漏洩で被害。
  対策として、データの匿名化など。

- 差別：
  ビッグデータで学習された人工知能が差別的な判断を下す。
  （例：Googleでの検索補完、ChatGPT ?!）

大規模データ処理における計算方法：

並列コンピューティング (PARALLEL COMPUTING)

分散コンピューティング (DISTRIBUTED COMPUTING)

- 大規模データを速く処理することが求められる。

- だがハードウェアの限界がある：

  一つのCPUで処理できるデータが限られている。

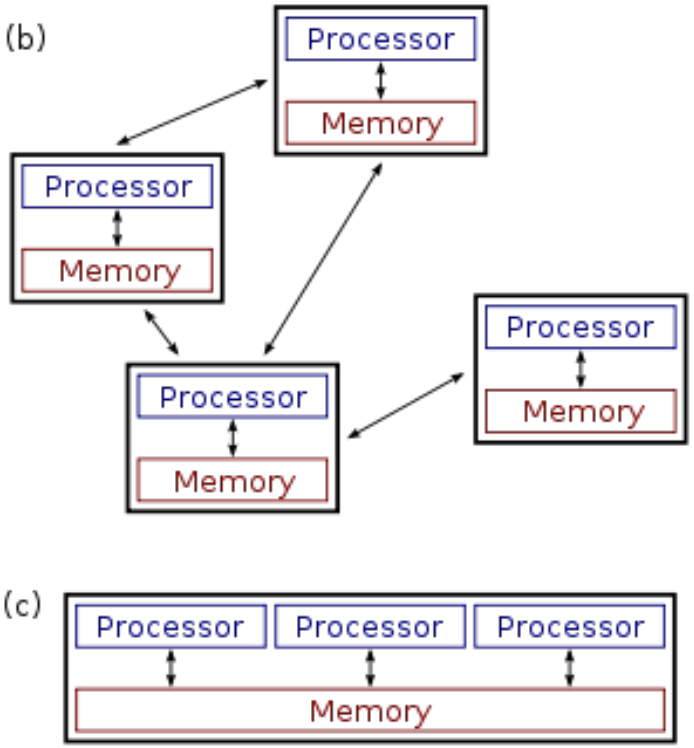
  一つのPCのメモリに全てのデータを格納できない。

- 基本的な対策：

  データを分割して処理する。以下の二つの方法がある。

  ・並列コンピューティング (Parallel Computing)

  ・分散コンピューティング (Distributed Computing)

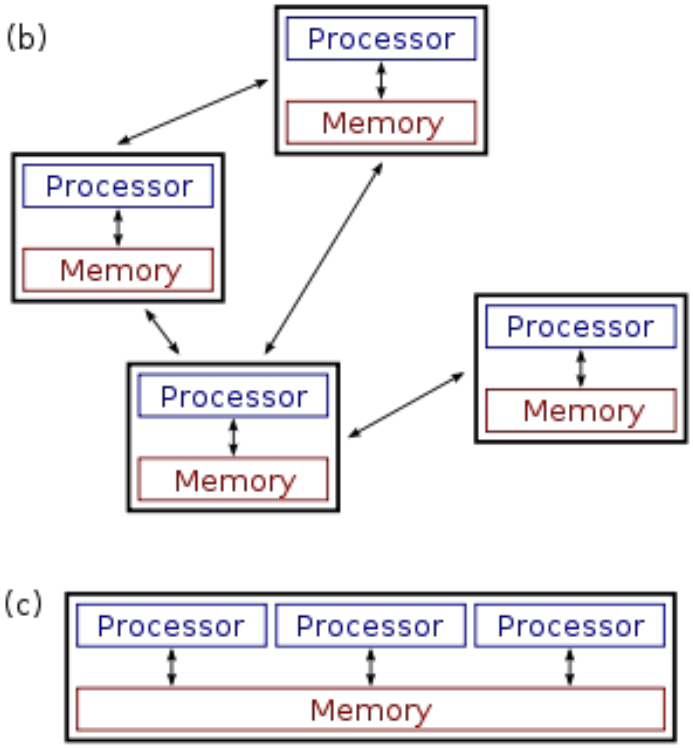


https://en.wikipedia.org/wiki/Distributed_computing より

並列コンピューティング (PARALLEL COMPUTING)
分散コンピューティング (DISTRIBUTED COMPUTING)

Distributed Computing
分散コンピューティングではメモリが分散されている。
例：**Spark**や**Hadoop**などがビグデータ処理に不可欠。

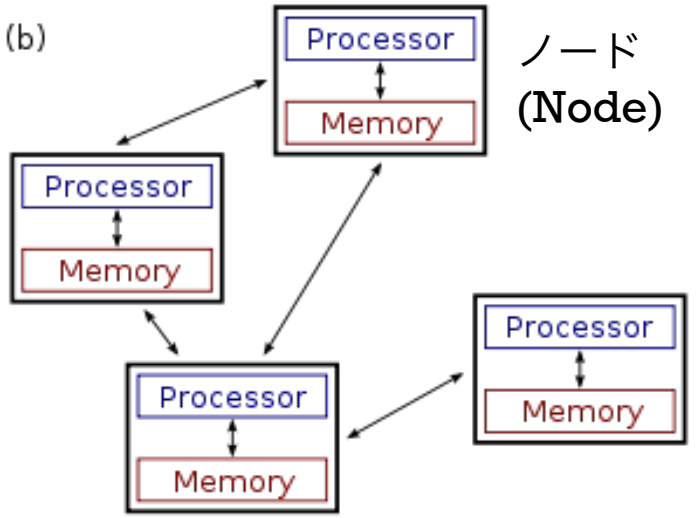
Parallel Computing
並列コンピューティングではメモリが共有されている。
例：**Numpy**が大きい行列を分解して複数の**CPU**で処理する。



https://en.wikipedia.org/wiki/Distributed_computing より

分散コンピューティング (DISTRIBUTED COMPUTING)

- Big Data処理に不可欠。
- ノード間の通信がボテルネックになっているから、その通信の量と頻度を抑えるが鍵。
- MapReduceは分散データ処理に利用された一つの代表的なアルゴリズムのフレームワーク



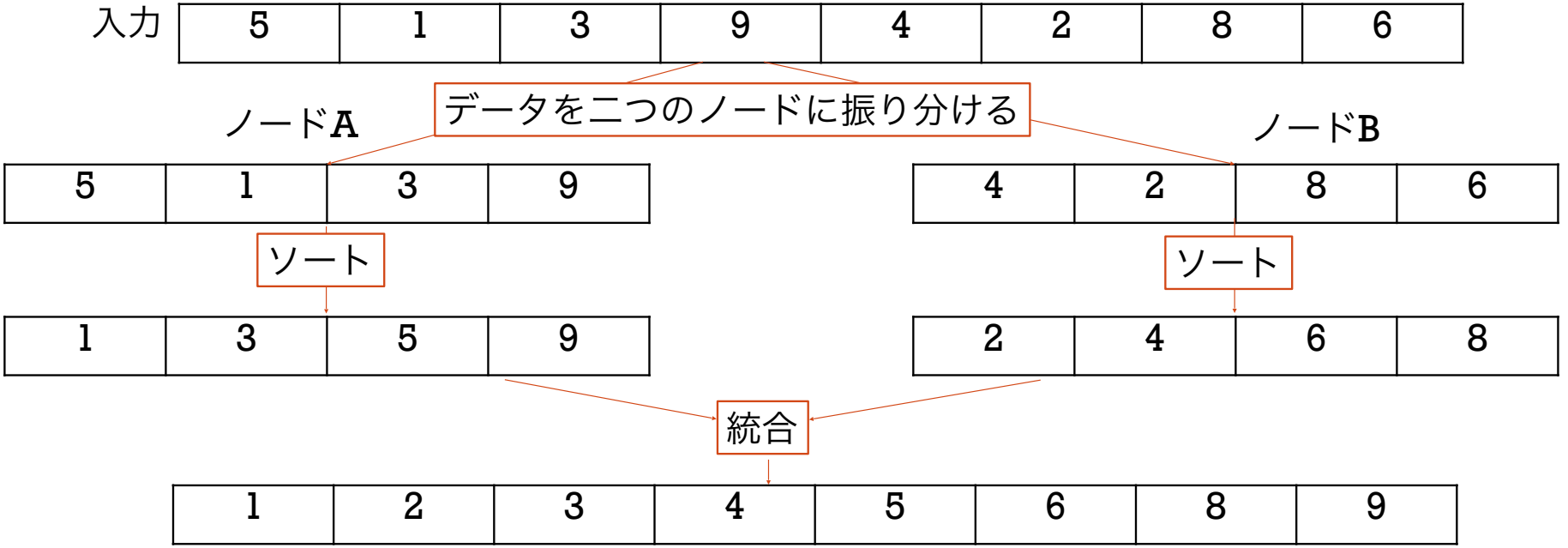
https://en.wikipedia.org/wiki/Distributed_computing より

並列処理とアルゴリズムの設計（1/2）

例：数字列のソート

- データのサイズはn, 計算コストがf(n) 、ノードの数はcとする。
- 計算コストがf(n/c)になるのが理想だが、ノード間の通信コストや各ノードの結果を集約する計算コストは無視できない。

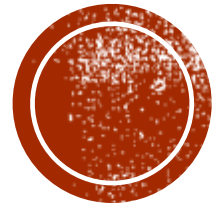
この例では**n = 8, c = 2:**



並列処理とアルゴリズムの設計（2/2）

- 従来のアルゴリズムや機械学習方法を並列処理に調節・再設計する必要がある。だが、場合によって、並列可能かつ効率の良いアルゴリズムの設計が自明ではない。
- 自明の場合では「Embarrassingly parallel」と呼ぶことが多い。例：行列の足し算（各要素を独立に足し算）
- 自明ではない例：
  - 逆行列を求める。
  - 深層学習（複数のノードを利用したHyperparameter検索は自明だが、モデルの学習にはに工夫が必要。）

質問タイム



4. 人工知能システムと学習データの重要性

人工知能　(ARTIFICIAL INTELLIGENCE)

従来型の人工知能：

- Expert System
  （専門家がルールを作成し、システムの知識ベースに加える）

現在の人工知能：

- データ駆動の機械学習（深層学習、ベイズモデルなどの統計モデルなど）。
- オンライン学習によって機械学習をビッグデータに適応。

逆に言えば、現在の人工知能には<u>データ</u>が不可欠。

データを四つの観点から考察

1. **構造の有無・種類:**

   Structured Data (構造化データ) 表（行列）として扱えるデータ。例：
   - 多肢選択のアンケート
   - 購入履歴

   Unstructured Data (非構造化データ) 前処理せずに表（行列）として扱いにくい。例：
   - 画像データ
   - センサーデータ
   - テキストデータ

2. **データの次元・量：**
   - n >> d: サンプル数が次元数より多い。例：ツイート, 大手企業の顧客データ
   - n << d, n = d: High-dimensional（高次元）データとも呼ぶ。例：遺伝子発現データ

3. **ラベル情報の有無：**
   - Labeled Data: ラベル付きデータ。例：２値分類の問題では正解かどうかの情報 (ツイートが誹謗に当たるかどうか）。
   - Unlabeled Data: ラベルなしのデータ。正解かどうかがわからない。

4. **アクセス・利用制限：**
   - 公開データ：自由に使ってよい。例：Wikipedia
   - 制限のあるデータ：Twitter, Facebook, Googleの検索キーワード履歴など。

現在の人工知能と学習データ

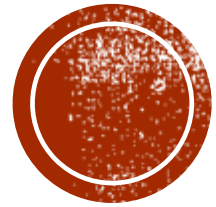
- 現在の人工知能では人工知能 = 機械学習 (Machine Learning)。
- 学習のためには、ラベル付きのデータが必要。(*)
- ビッグデータとはいえラベル付きのデータが少ない場合が多い。
- 例：
  - SNSにおける誹謗判別データ。
  - インターネット上の画像データにおける顔認証。

(*) ラベルなしの学習方法もあるが、応用が限られている。

(**) ラベルを付けることは「Annotation」と呼ぶことが多い。

- ラベル付きのデータが少ない問題の対策：
  - **Crowdsourcing**: 低コストにラベルを付与。(**)
    例：Amazon Mechanical Turk。
  - **Active-Learning**（能動学習）：効率よくラベルを付与。
    例：識別平面に近いサンプルを選んで、ラベルを付与する。
  - **Semi-Supervised Learning**（半教師あり学習）： ラベルなしのデータを生かす。
    例：Word Embedding、Transformer

質問タイム



5. 情報・人工知能システムにおける安心・安全性

情報システムの安心・安全性 (1/4)

安心・安全性を次の観点から考察できる。

- **Safety** （安全性）・ **Security** （セキュリティ）
- **Reliability** (信頼性) ・ **Availability** 可用性 ・ **Maintainability** (保守性)
- **Risk Classification** (リスクの分類)

情報システムの安心・安全性 (2/4)

Safety （安全性）・　Security （セキュリティ）

- **Safety （安全性）**：
  システムの誤操作やバグによる被害を防ぐ、または最小限にする。例えば、**ETC**(有料道路料金自動徴収システム）の誤操作によって遮断棒が突然に下がることを防ぐ。

- **Security （セキュリティ）**：
  システムやシステムの扱っているデータが悪用されるからの防衛。例えば、サイバー攻撃によって情報が盗まれたりすることを防ぐ。

情報システムの安心・安全性 (3/4)

情報システムにおける**RAM**：

- 信頼性（**R**eliability)

  システムの動作によって要件が満たされているか。

  「ユーザに影響するエラーを防ぎ、目的の機能が十分に実現されていることを意味する。」[1]

- 可用性 (**A**vailability)

  システムがそもそも動作しているか。例：

  - ネットワーク障害などによって、アクセスできない状態を最小限。
  - 災害対策。顧客のデータを東京にあるサーバだけでバックアップするのはどうなのか。

- 保守性 (**M**aintainability)

  - システムの修正が容易なのか。
  - 脆弱性の修正ソフトウェア（パッチ）の開発が容易にできるか。

    ソフトウェア工学の観点から見ると、ソースコードを簡潔・分かりやすく書いたり、

    Refactoringしたりといった方法でソースコードの保守性を高める。

[1] https://ja.wikipedia.org/wiki/ソフトウェア品質　https://en.wikipedia.org/wiki/Reliability,_availability_and_serviceability

情報システムの安心・安全性 (4/4)

情報システムにおけるリスクの分類：

	区分	例
安全性	災害	地震、落雷、洪水、火事、停電、漏水
安全性	故障	システム障害、ネットワーク障害、設備障害
安全性	過失	入力ミス、操作ミス、運用ミス、チェック漏れ、ソフトウェアバグ
セキュリティ	不正	物理的破壊、ネットワーク不正アクセス、権限外使用、情報の盗用・漏えい・改ざん

表4 情報システムに対する「脅威」

https://www.itmedia.co.jp/im/articles/0403/27/news017.html より

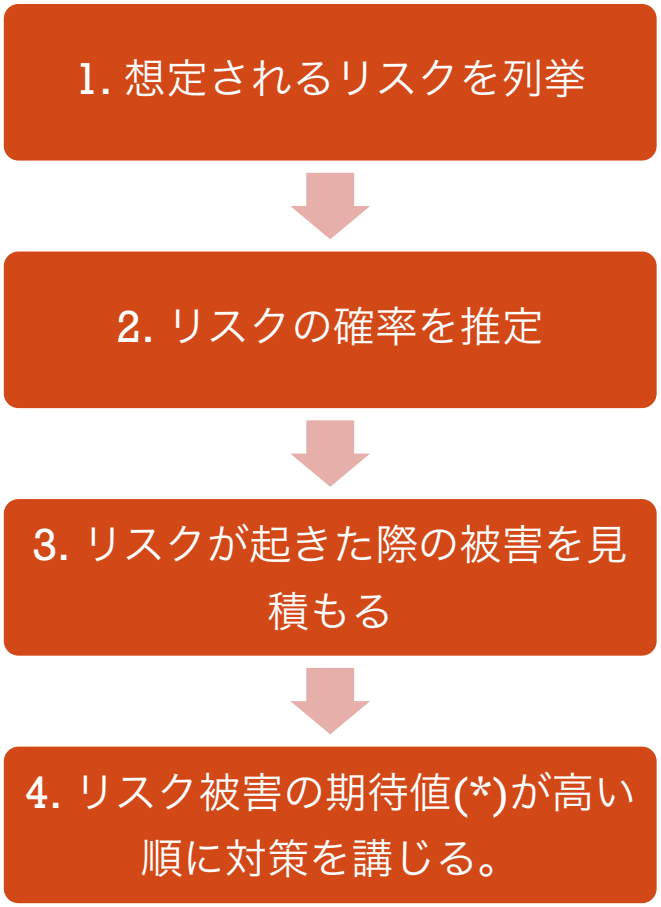
情報システムの安心・安全性を強化 (1/3)

安心・安全性を強化するために、次の方法を紹介する。

- リスク分析
- ソフトウェアとハードウェアの管理

情報システムの安心・安全性を強化 (2/3)

リスク分析



(*) リスク発生確率と損害の積

情報システムの安心・安全性を強化 (3/3)

ソフトウェアとハードウェアの管理

- ソフトウェアの品質管理：
  - 航空コントロルといった安全重視のシステムではModel Checkingも利用されている。（プログラムの理論的な保証が可能）
  - 二重化：同じ機能が二つの独立のチームによって実装される。

- ハードウェアの管理：
  - 冗長化, 例えば、ハードディスクのバックアップ, RAID (Redundant Arrays of Inexpensive Disks）
  - CPUの温度センサーやクーラ

人工知能システムの安心・安全性とその強化

注意点は一般的な情報システムと同じ点が多いが、人工知能システムに特有な点もある。

これから以下を紹介する。

- 人工知能に特に注意すべきリスク・脆弱
- 人工知能の安心・安全性を強化するための方法

人工知能に特に注意すべきリスク・脆弱

- **安全性に関するリスク：**
  人工知能の間違った判断による損傷を防ぐ。（システムの設計者とユーザの意図と違う判断）

- **セキュリティに関する脆弱：**
  Adversarial Attackによって人工知能システムを騙して、意図的に間違って判断させる。

- **信頼性に関するリスク：**
  従来の情報システムでは設計上で性能の担保は可能だが、機械学習では困難。特にNNのような複雑な機械学習モデルではどのような条件でどのような結果が出るかの約束が困難。

- **保守性に関するリスク：**
  環境の変化によって人工知能システムの性能が劣化。学習データにない現象に対応が困難。例：需要予測のシステムは新商品の導入によって予測性能が劣化。

- **公平性に関するリスク：**
  従来の情報システム開発にはあまり考えられていない。
  従来では設計者がコントロールできたが、今では「学習データ」によって勝手にバイアスや差別の問題が起こりうる。例：Google検索の補完, 個人データから犯罪者の分類。

人工知能システムの安心・安全性を強化する方法

(*) Explainability （説明性）ともいう。

- **Interpretability （解釈性・解釈可能性）(*)：**
  なぜこの判別結果になったのか説明できる材料を示す。
  NNのような複雑な機械学習モデルでは判別性能が高いが、解釈性が低い。
  一方、Logistic Regression (ロジスティック回帰)や Decision Tree (決定木）では解釈性が高いが、
  性能がよくない場合がある。
  解釈性と判別性能のトレードオフはあるが、
  高い判別性能を保ちながら、よい解釈性・説明性を持つのが現在に盛んでいる研究領域の一つ。

- **Robustness (ロバスト統計・学習方法)：**
  外れ値（異常のサンプルや敵対サンプル, Data Poisoning）にロバスト(頑健性）な学習方法やモデルを利用。
  例：L2損失の代わりにHuber損失を利用。

- **Uncertainty Quantification（確率的な評価）：**
  人工知能システムが予測・判別結果を出力するだけではなく、システムが持っている確信も合わせて出力する。
  例：赤信号か青信号の分類問題　出力：(赤信号, 55%確率) vs （赤信号, 99%)。前者の場合ではユーザに相談すべき、後者の場合は人工知能に指導権を渡すことも可能。

人工知能システムの安心・安全性・倫理

人工知能における安心・安全性・倫理に関する詳細は以下の文献をお勧めします。

“Understanding artificial intelligence ethics and safety - A guide for the responsible design and implementation of AI systems in the public sector”, The Alan Turing Institute
https://www.turing.ac.uk/sites/default/files/2019-06/understanding_artificial_intelligence_ethics_and_safety.pdf

Safetyに関する詳細：

p30 – p34.

Interpretabilityに関する詳細：

38p以降

本日のまとめ

- 情報システムの基本構成には集中型（主な処理がホストで行われている）と分散型（情報処理の分担が複数の計算機に分散される）がある。
- クラウドコンピューティングは集中型の側面もあり、分散型の側面もある。クラウドコンピューティンにおいて、経済的なメリットが多くて、幅広く普及しつつある。
- 一つのCPUで計算できる限界を乗り越えるために、並列計算と分散計算が利用されている。効率的な並列・分散処理を実現するためには場合によって工夫が必要。
  ビッグデータの処理には分散計算が不可欠。
- 現在の人工知能（＝機械学習）には学習用のデータが不可欠。
- 情報システム・人工知能システムでは特に以下の必要条件があげられている：
  安全性, セキュリティ, 信頼性, 保守性, 公平性。
  それらを満たすために、人工知能の解釈性(Interpretability)を高める方法が近年に注目されている。

レポートの課題

（BLACKBOARDに登録されている）

(1)　人工知能システムにおけるセキュリティ

[1]のp32,p33で定義したAdversarial AttackとData Poisoningを読んで、 Data Poisoningの例を一つを書いてください。

(2)　人工知能における解釈性（ Interpretable AI）：

解釈性のある人工知能システムの利点を二つ以上あげてください。[1]のp38,p39を参考にしてください。

[1] “Understanding artificial intelligence ethics and safety - A guide for the responsible design and implementation of AI systems in the public sector”, The Alan Turing Institute
https://www.turing.ac.uk/sites/default/files/2019-06/understanding_artificial_intelligence_ethics_and_safety.pdf

レポートの課題

- 課題（１）と（２）どちらも回答してください。

- 締め切りは4月30日（日曜日）の23時55分です。
  締切までは再提出可で，最後に提出されたものを採点します。

- レポートに関する質問は4月29日までにお願いします。
  （締め切りの当日に私はメール・Slackのメッセージに返事できない可能性があるため。）

- 解答の方法：
  - Blackboardに登録
  - 最初の一行や一段落で、解答を簡潔に書いてください。
    その後に、解答に至るまでの考え方・補足・詳細を簡潔に書いてください。
  - Docxファイルとして提出しないでください。pdfに変換したもののご提出をお願いします。

補足

補足：CLIENT/SERVER

- クライアント：
  - 情報を要求する
  - 例：Web Browser

- サーバ：
  - 情報を提供する
  - 例：インターネットの向こうにある高性能のパソコン

例：

クライアント(Client)

サービスを要求する側

サーバ(Server), ホスト(Host)
サービスを提供している側

補足：MAPREDUCE

- マスターノードがデータを分割している。
- Mapステップ担当のノード：
  - 分割したデータを処理して、その結果とキーと共に出力。
- Reduceステップ担当のノード：
  - キーを元にMapノードからの結果を集約する。

- 各MapとReduceノードが独立に処理を行える。
- MapReduceはメタの戦略だけであり、データ処理目的によって、具体的にMapとReduceのステップを定義する必要がある。