令和2年度「専修学校による地域産業中核的人材養成事業」

人工知能概論

令和2年度「専修学校による地域産業中核的人材養成事業」

人工知能概論

Society5.0 実現のための IT 技術者養成モデルカリキュラム開発と実証事業

目次

AIシステム科　　シラバス　　20XX年度入学生

科目名	人工知能概論				週合計駒数	1駒	作成日	
区分	必修 講義	開講時期	1年次 前期	週講義駒数 週実習等駒数	1駒 0駒	総時間数 総単位数	30時間 2単位	担当教員	

目標	概要
これまでの人工知能の発展を知るとともに、人工知能に関する各種問題、人工知能の種類、基本的な人工知能アルゴリズムの習得を目標とする。	これまでの人工知能研究の成果や問題点、各種アルゴリズムを学ぶことで、人工知能研究の目的について学習する。

履修前提 ※選択・エクステンションのみ記入	テキスト・参考文献
	オリジナルテキスト

評価方法	関連科目
小テスト／中間テスト／期末テスト、提出課題、授業に取り組む姿勢（出席率、授業態度）	AIプログラミングⅠ・Ⅱ・Ⅲ、機械学習Ⅰ・Ⅱ・Ⅲ、人工知能特論、AIシステム開発

1	学習目標 現在の人工知能を取り巻く状況を説明出来る。	学習項目 人工知能の歴史と発展を踏まえた上で、現在の人工知能の状況について学習する。最近のAI関連のトピックについても学習する。
	理解度確認：練習問題、小テスト	
2	学習目標 人工知能の各種問題（一般化フレーム問題、シンボルグラウンディング問題、チューリングテスト、など）を説明出来る。	学習項目 人工知能に関する根本的な各種問題（一般化フレーム問題、シンボルグラウンディング問題、チューリングテスト、など）について学習する。
	理解度確認：練習問題、小テスト	
3	学習目標 ルールベースの人工知能について説明出来る。知識ベースの人工知能について説明出来る。エキスパートシステムについて説明出来る。	学習項目 決定木とは何かを理解した上で、ルールベースの人工知能、知識ベースの人工知能、エキスパートシステムについて学習する。
	理解度確認：練習問題、小テスト	
4	学習目標 画像処理について説明出来る。OCR、物体認識について説明出来る。	学習項目 画像をどのようにデジタルで取り扱うかを理解した上で、OCRについて学習する。併せて物体認識についても学習する。
	理解度確認：練習問題、小テスト	
5	学習目標 パターン検索について説明出来る。	学習項目 Eメール本文からの電話番号抽出を題材にして、パターン検索について学習する。
	理解度確認：練習問題、小テスト	
6	学習目標 これまでに学習した内容を復習し、理解を確実なものにする。	学習項目 これまでに学習した内容の理解を確実にするため、総復習を行う。
	理解度確認：演習問題	
7	学習目標 ゲーム理論について説明出来る。標準型ゲームと展開型ゲームについて説明出来る。	学習項目 ゲーム理論について学習する。「囚人のジレンマ」で考え方について学んだ後、宿泊施設における部屋の価格設定問題への応用を題材にして学習する。
	理解度確認：練習問題、小テスト	
8	学習目標 グラフ理論とは何かについて説明出来る。グラフ探索（ミニマックス、ネガマックス）について説明出来る。	学習項目 グラフ理論について学習する。ここではグラフ探索のミニマックスとネガマックスについて学習する。
	理解度確認：練習問題、小テスト	
9	学習目標 アルファベータ法について説明出来る。	学習項目 グラフ理論について学習する。ここではグラフ探索のアルファベータ法について学習する。
	理解度確認：練習問題、小テスト	
10	学習目標 グラフ探索（深さ優先探索、幅優先探索、A*探索）について説明出来る。	学習項目 グラフ探索のうち、深さ優先探索、幅優先探索、A*探索について学習する。動的計画法についても学習する。
	理解度確認：練習問題、小テスト	
11	学習目標 遺伝的アルゴリズムとは何かを説明出来る。巡回セールスマン問題について説明出来る。	学習項目 遺伝的アルゴリズムについて学習する。ここでは巡回セールスマン問題について学習する。
	理解度確認：練習問題、小テスト	
12	学習目標 これまでに学習した内容を復習し、理解を確実なものにする。	学習項目 これまでに学習した内容の理解を確実にするため、総復習を行う。
	理解度確認：演習問題	
13	学習目標 自然言語処理とは何かを説明出来る。構文解析について説明出来る。	学習項目 自然言語処理とは何かを踏まえた上で、文章の構造と理解、構文解析について学習する。
	理解度確認：練習問題、小テスト	
14	学習目標 自然言語処理とは何かを説明出来る。構文解析について説明出来る。	学習項目 自然言語処理の代表的な構文解析のアルゴリズム（チャート法、Cocke-Younger-Kasami法、など）について学習する。
	理解度確認：練習問題、小テスト	
15	学習目標 これまでに学習した内容を復習し、理解を確実なものにする。	学習項目 これまでに学習した内容の理解を確実にするため、総復習を行う。
	理解度確認：演習問題	

第1回：人工知能を取り巻く状況

全15回の講義について

- これまでの人工知能の発展を知るとともに、人工知能に関する各種問題、人工知能の種類、基本的な人工知能アルゴリズムの習得を目標とします。

アジェンダ

- 人工知能の歴史
- 最近のAIに関するトピック

人工知能の歴史

人工知能の歴史

人工知能学会の記事[https://www.ai-gakkai.or.jp/whatsai/AIhistory.html]を参考にしています。

この記事の中では、人工知能の歴史を5つに分けて記述しています。

1. 人工知能の夜明け（～1956）
2. 古き良き人工知能（1957～1969）
3. 現実からの反撃（1970～1979）
4. 人工知能の産業化（1980～1988）
5. 現在そして未来の彼方へ（1989～）

人工知能の夜明け （～1956）

1943 W.McCulloch（ウォーレン・マカロック）とW.Pitts（ウォルター・ピッツ）が“A Logical Calculus of the Ideas Immanent in Nervous Activity”を出版. ニューラルネットワークの基礎となりました.

1943 A.Rosenblueth（ローゼンブリュート）, N.Wiener（ウィーナー ）, J.Bigelow（ビゲロー）が論文で“サイバネティクス”という言葉を用いました.

- サイバネティクスとは、機械の自動制御や動物の神経系機能の類似性や関連性をテーマに研究する、心理学、生物学、物理学、数学等を包括した科学の総称です。
- アメリカの情報理論の大家であるノーバート・ウィーナー（Norbert Wiener）が、1948年に初めて発表したサイバネティクス理論は、生物と機械の間に情報のやりとりやコントロールの仕組みなどに関する類似性があることに着目し、自然から人工機構まで含めた多種多様な学問領域が協同して取り組む新しい研究課題への道を切り開きました。

人工知能の夜明け（～1956）

1950 A.M.Turingが“Computing Machiery and Intelligence”を出版．知的活動をテストする方法としてチューリングテストを示しました．

- 「機械が思考したかどうかは、人との会話が成立したかどうかで判断する。」と定義したテストです。人の審査員が、相手が見えない状況で「人」もしくは「コンピュータ」と対話し、相手が人なのかプログラムなのかを言い当てました。審査員がプログラムと対話した後に「『人と』対話した」と間違った答えを出せば、「機械は思考した」ということにしました。

1950 I.Asimov がロボット3原則を発表しました．「人間を傷つけてはななならい．傷つくのを看過してはならない」「第1原則に反しない限り，人間の命令に従わなくてはならない」「第1，第2原則に反しない限り自分の身を守らなくてはならない」

古き良き人工知能（1957～1969）

1957 J.Backus（ジョン・バッカス）が最初の高級言語FORTRANを開発．

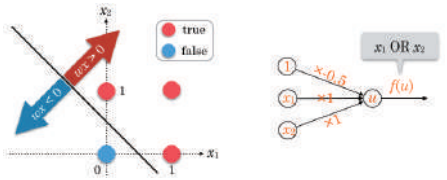
1965 J.Weizenbaum（ジョセフ・ワイゼンバウム）がELIZA（イライザ）を開発．英語でいろいろな話題について会話ができるプログラムで，精神科医をまねたバージョンはネットワーク上で人気を集めた．

- イライザは「考えて回答」しているわけではなく、会話のパターンを利用していました。例えば人がイライザに「お腹が痛い」と言えば、イライザは「なぜお腹が痛いのか？」と返すようになっていました。仕込んだパターン以外の質問をイライザにすると、イライザは回答できませんでした。

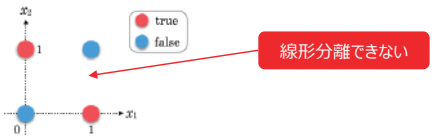
古き良き人工知能（1957～1969）

1968 M.MinskyとS.PapertがPerceptronsを出版し単層ニューラルネットであるパーセプトロンの限界を指摘.

単純パーセプトロンでは線形非分離な問題が解決できない。



OR（論理和）の例



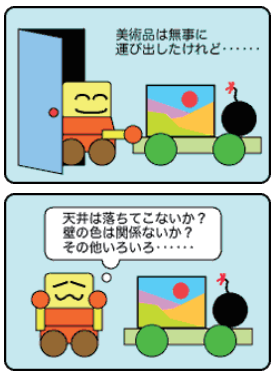
NAND（否定論理積）の例

出展： http://hokuts.com/2015/12/04/ml3-mlp/

古き良き人工知能（1957～1969）

1969 J.McCarthyとP.J.Hayesが人工知能最大の難問“フレーム問題”を指摘.

[人工知能学会HPより]
フレーム問題は，今からしようとしていることに関係のあることがらだけを選び出すことが，実は非常に難しいという問題です.
人工知能搭載のロボット「安全くん1号」は，人間の代わりに危険な作業をするロボットです．爆弾が仕掛けられている部屋から貴重な美術品を取り出してこなければなりません．安全くん1号は美術品の入った台車を押して美術品をとってきましたが，不幸なことに爆弾は台車にしかけられていたので，安全くんは爆発に巻き込まれてしまいました.
これは安全くん1号が，美術品を取り出すために荷車を押せばよいということは分かったのですが，そのことによって，爆弾も一緒に取り出してしまうということは分からなかったためでした.

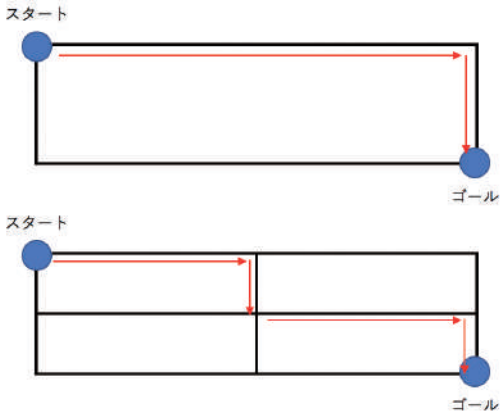


出展（人工知能学会）：http://hokuts.com/2015/12/04/ml3-mlp/

現実からの反撃（1970～1979）

1973 ライトヒル勧告．組合せ爆発問題を指摘した勧告．イギリスで2大学を除きAI研究の補助金がうち切られる．

「格子状の道をスタート地点からゴールまで同じところを2度と通らない道順の数」は、格子の数が増えると爆発的に増えます。



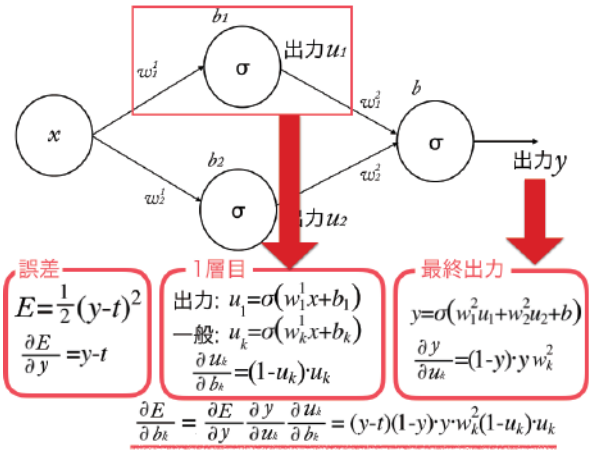
人工知能の産業化（1980～1988）

1982 日本で第5世代プロジェクトの開始．超並列で論理型言語を実行するコンピュータと自然言語の理解などを目標としました．

[Wikipediaより]
1980年代に入り、日本のコンピュータ産業は輸出も増え、市場規模も2兆円まで成長した。従来、通産省は1983年ごろまで貿易自由化対策としてコンピュータ企業への助成金を出していたが、既にそのような直接的な助成金は意義を失っていた。また、海外からもIBM互換機を輸出する日本に対して風当たりが強くなっていた時期でもある（IBM産業スパイ事件が起きたのは1982年）。そこで、次は第四世代と言われていた時代に、あえて更に先の第五世代コンピュータを開発するプロジェクトを立ち上げ、日本の独自性を打ち出そうとした。
この検討が開始されたのが1979年である。当時、電子技術総合研究所（現在の産業技術総合研究所）の渕一博らは述語論理によるプログラミングに強い関心を持っていた。渕らは独創性を求めるこのプロジェクトを絶好の機会として働きかけ、第五世代コンピュータの目標は「述語論理による推論を高速実行する並列推論マシンとそのオペレーティングシステムを構築する」というものになった。
当初の予定から1年延びた1992年、プロジェクトは「当初の目標を達成した」として完了した。

人工知能の産業化（1980～1988）

1980中 ニューラルネットのバックプロパゲーション・アルゴリズムが広く用いられるようになる.

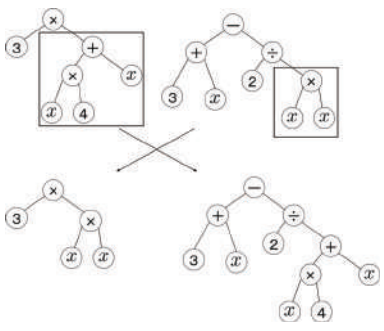


出展： https://www.yukisako.xyz/entry/backpropagation

現在そして未来の彼方へ（1989～）

1990 J.R.Kozaが遺伝的プログラミングを開始.

[Wikipediaより]
遺伝的プログラミングは1990年にジョン・コザ（John Koza）によって提案された。他の進化的アルゴリズムの主要な方法論が同時期に提案され独立して研究が進められていたのに対し、遺伝的プログラミングは最初から遺伝的アルゴリズムの拡張として提案されており、他の三つの方法とは大きく立場を異にする。



現在そして未来の彼方へ（1989～）

1990中 データマイニング技術の誕生.

- データマイニング（Data mining）とは、統計学、パターン認識、人工知能等のデータ解析の技法を大量のデータに網羅的に適用することで知識を取り出す技術・プロセスの総称です。
- 英語では“Data mining”の語の直接の起源となった研究分野であるknowledge-discovery in databases（データベースからの知識発見）の頭文字をとってKDDとも呼ばれます。

最近のAIに関するトピック

人事分野への応用

- 自然言語処理により、応募者が記載したエントリーシートの内容を評価する事例です。
- 人事担当者の負担の軽減、評価基準の統一などの効果があります。

プレスリリース 2017年

いいね！ ツイート　一覧へ戻る | ＜前のリリース | 次のリリース＞

新卒採用選考におけるIBM Watsonの活用について

2017年5月29日
ソフトバンク株式会社

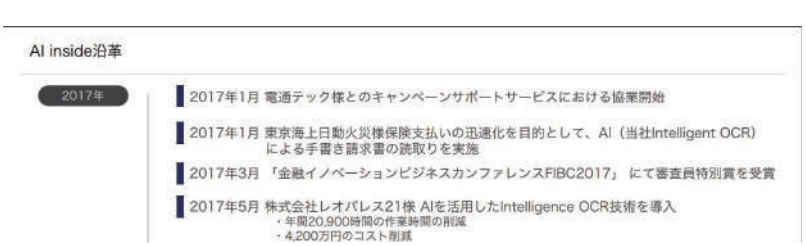
ソフトバンク株式会社は、応募者をより客観的に、また適正に評価することを目的に、2017年5月29日より新卒採用選考のエントリーシート※評価にIBM Watson日本語版（以下「IBM Watson」）を活用します。

過去のデータを学習させたIBM Watsonに応募者のエントリーシートデータを読み込ませると、IBM WatsonのAPIの一つであるNLC（Natural Language Classifier、自然言語分類）により、エントリーシートの内容が認識され、項目ごとに評価が提示されます。合格基準を満たす評価が提示された項目については、選考通過とし、それ以外の項目については人事担当者が内容を確認し、合否の最終判断を行います。IBM Watsonによる評価をエントリーシート選考の合否判断に活用することで、統一された評価軸でのより公平な選考を目指します。

出展：https://www.softbank.jp/corp/group/sbm/news/press/2017/20170529_01/

業務効率化

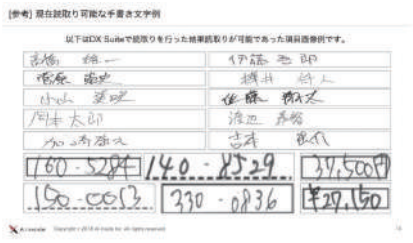
- OCR（Optical Character Recognition：光学的文字認識）により手書き文字をAIが認識する事例です。
- 事務作業の効率化ができます。



AI inside沿革

2017年

- 2017年1月 電通テック様とのキャンペーンサポートサービスにおける協業開始
- 2017年1月 東京海上日動火災様保険支払いの迅速化を目的として、AI（当社Intelligent OCR）による手書き請求書の読取りを実施
- 2017年3月 「金融イノベーションビジネスカンファレンスFIBC2017」 にて審査員特別賞を受賞
- 2017年5月 株式会社レオパレス21様 AIを活用したIntelligence OCR技術を導入
  - 年間20,900時間の作業時間の削減
  - 4,200万円のコスト削減



出展：https://rpa-bank.com/pdf/document-dl/ai_inside.pdf

新商品開発

- 保険会社が新商品開発にAIを使用した事例です。
- 保険会社は新しい商品の売上が増え、保険を契約する顧客は機器の保守コストを抑えることができる事例です。

ニュース

東京海上日動と日立が保険サービスを共同開発、IoTとAIで予兆診断

山端 宏実＝日経 xTECH／日経コンピュータ　日経 XTECH

この記事を評価する

この記事は　仕事に役立った0　人に勧めたい0　難しい0　易しい0

東京海上日動火災保険と日立製作所は2019年1月16日、データを活用した保険サービスを共同開発することで合意したと発表した。東京海上日動が持つ事故データや保険サービスと、日立のIoT（インターネット・オブ・シングズ）やAI（人工知能）などの技術を組み合わせ、新たな保険サービスを生み出す狙いだ。

第1弾として月内に、日立のIoTやAIによる予兆診断技術を活用し、製造設備の異常の兆候をつかみ、検査などに必要な費用を補償する保険の提供を始める。従来の保険は物的損壊を補償の要件にしていた。設備の異常の兆候を検知し、あらかじめ対策を講じることができれば企業は損失を最小限に抑えられ、保険会社が支払う補償額も少なく済む利点がある。

出展：https://tech.nikkeibp.co.jp/atcl/nxt/news/18/03854/

販促活動

- アサヒビールが、自社製品を販売してくれる小売企業（スーパー）に対して値付けサービスを提供するという事例です。
- アサヒビールを取り扱う小売企業が潤えば、アサヒビールも潤うというサプライチェーン全体を見据えた取り組みの事例です。

アサヒ、AIでビール売価指南　販売てこ入れ

2018/8/21 20:58 | 日本経済新聞　電子版

保存　共有　印刷　その他

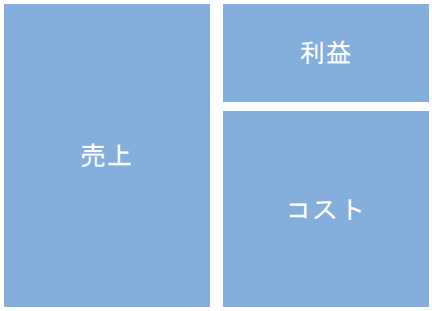
アサヒビールは、スーパーなどがビール系飲料の収益を効率良く上げるため、最適な値付けを提案できるシステムを売り込む。人工知能（AI）が販売環境などを踏まえ予測する。2017年6月の酒の安売り規制強化後、販売競争は激しい。小売店はライバル店の動きを横目に原価を下回らずに値ごろ感をどう出すか悩んでいる。競争力のある価格設定を手助けし、小売りとの取引拡大につなげる。

システムはNECのAI技術を組み込み、同社と共同開発した。小売店からPOS（販売時点情報管理）データに加え、天気や気温、運動会といった周辺イベント情報の提供を受けAIに入力する。入力データをもとに小売店の売り上げや利益を効率良くあげるにはどんな商品を、どの時期にどんな価格で売ればよいかAIがはじき出す。

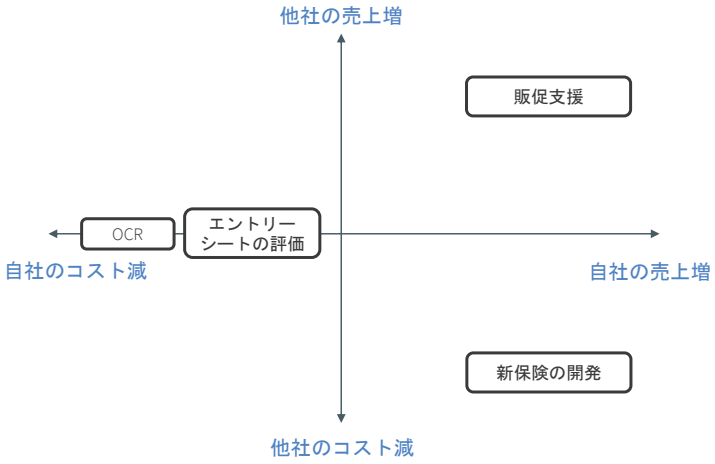
出展：https://www.nikkei.com/article/DGXMZO34407380210820189l6M00/

AIをどこで使うか

- 企業活動においてAIを適用する効果として①売上が増えるのか、もしくは②コストが減るのかのどちらかに大別できます。



企業における収支のバランス



第2回：人工知能の各種問題

本講義について

- 人工知能に関する根本的な各種問題（一般化フレーム問題、シンボルグラウンディング問題、チューリングテスト、など）について学習します 。

アジェンダ

- 一般化フレーム問題
- シンボルグラウンディング問題
- チューリングテスト

一般化フレーム問題

フレーム問題と一般化フレーム問題

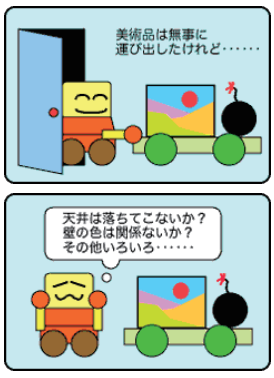
[Wikipediaより]
フレーム問題（フレームもんだい、英: Frame problem）とは、人工知能における重要な難問の一つで、有限の情報処理能力しかないロボットには、現実に起こりうる問題全てに対処することができないことを示すものである。1969年、ジョン・マッカーシーとパトリック・ヘイズ（英語版）の論文[1]の中で述べられたのが最初で、現在では、数多くの定式化がある。

現実世界で人工知能が、たとえば「マクドナルドでハンバーガーを買え」のような問題を解くことを要求されたとする。現実世界では無数の出来事が起きる可能性があるが、そのほとんどは当面の問題と関係ない。人工知能は起こりうる出来事の中から、「マクドナルドのハンバーガーを買う」に関連することだけを振るい分けて抽出し、それ以外の事柄に関して当面無視して思考しなければならない。全てを考慮すると無限の時間がかかってしまうからである。つまり、枠（フレーム）を作って、その枠の中だけで思考する。
だが、一つの可能性が当面の問題と関係するかどうかをどれだけ高速のコンピュータで評価しても、振るい分けをしなければならない可能性が無数にあるため、抽出する段階で無限の時間がかかってしまう。
これがフレーム問題である。
あらかじめフレームを複数定義しておき、状況に応じて適切なフレームを選択して使えば解決できるように思えるが、どのフレームを現在の状況に適用すべきか評価する時点で同じ問題が発生する。

フレーム問題と一般化フレーム問題

1969 J.McCarthyとP.J.Hayesが人工知能最大の難問“フレーム問題”を指摘.

[人工知能学会HPより]
フレーム問題は，今からしようとしていることに関係のあることがらだけを選び出すことが，実は非常に難しいという問題です．
人工知能搭載のロボット「安全くん1号」は，人間の代わりに危険な作業をするロボットです．爆弾が仕掛けられている部屋から貴重な美術品を取り出してこなければなりません．安全くん1号は美術品の入った台車を押して美術品をとってきましたが，不幸なことに爆弾は台車にしかけられていたので，安全くんは爆発に巻き込まれてしまいました．
これは安全くん1号が，美術品を取り出すために荷車を押せばよいということは分かったのですが，そのことによって，爆弾も一緒に取り出してしまうということは分からなかったためでした．



出展（人工知能学会）：http://hokuts.com/2015/12/04/ml3-mlp/

フレーム問題と一般化フレーム問題

フレーム問題が問題としているのは、考慮すべき空間が有限でない限り、無限の可能性について考えざるを得ないという点である（ただし、空間が有限でも、考慮すべき要素の組み合わせが爆発的に増加するので同じことである）。

自然界に発生した知性（人間の知性など）が、どのようにこのフレーム問題を解決しているかはまだ解明されていない。人間は実際にはフレーム問題を解決できておらず、フレーム問題にうまく対処しているかのように見えるだけだと唱える研究者もいる。この場合、どのように振舞わせれば、そう見せかけられるのかが研究の主題となる。このような、人工知能だけに限らず人間の知能にも起こり得るフレーム問題は、ジョン・マッカーシーらの提案したフレーム問題と区別して一般化フレーム問題と呼ばれている。

人間が自然に実現しているフレームの例

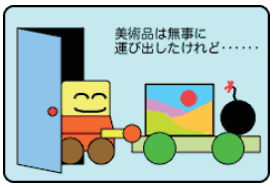
日常生活において、人間は滅多にフレーム問題に悩まされません。
これは人間が有限サイズのフレームで情報を囲っているからだと考えられます。つまり、無限の情報を処理する必要はなく、そのフレーム内の情報のみを処理すればよい、ということです。

人間がフレーム問題をうまく処理している例を挙げます。

- 「いつものところに6時」
  当事者間に共通の認識がある場合、これほど曖昧でも約束は成立します。
  つまり情報がフレームで囲われているため、指示対象が一つに決定しています。
- 「他人に迷惑をかけるな」
  当事者同士がどのような状況に置かれているのか、何を指して発言しているのか共通認識がある場合、これが成立します。
  フレームで情報を囲わないと、考慮するべき対象が無限に広がってしまい、処理能力が足りずにパンクします。

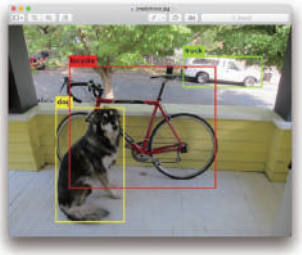
現代の技術で何が解決できそうか

[人工知能学会HPより]
安全くん1号は美術品の入った台車を押して美術品を
とってきましたが，不幸なことに爆弾は台車にしかけられていたので，安全くんは
爆発に巻き込まれてしまいました.



出展（人工知能学会）：
http://hokuts.com/2015/12/04/ml3-mlp/

このような問題は、例えば画像認識（一般物体認識）と[if else]のロジックで解決できる部分が多々あると思われます。
まず、一般物体認識で美術品とそれ以外の物体を判別します。
次に、「美術品以外の物体を認識した場合は、それを排除する」というif else ロジックを実行します。



出展（YOLO）：
https://pjreddie.com/darknet/yolo/

演習：フレーム問題

フレーム問題に対して、現在の技術を使って何が解決できそうか、議論してください。

シンボルグラウンディング問題

シンボルグラウンディング問題とは

[Wikipediaより]
シンボルグラウンディング問題（シンボルグラウンディングもんだい）とは、記号システム内のシンボルがどのようにして実世界の意味と結びつけられるかという問題。記号接地問題とも言う。ハルナッド(w:Stevan Harnad)によって命名された。

例えば「雪」を一度も見たことがない南国の人は、「雪」という記号（シンボル：文字列）を与えられただけでは「雪」が一体何なのかを理解できません（グラウンディングできない）。

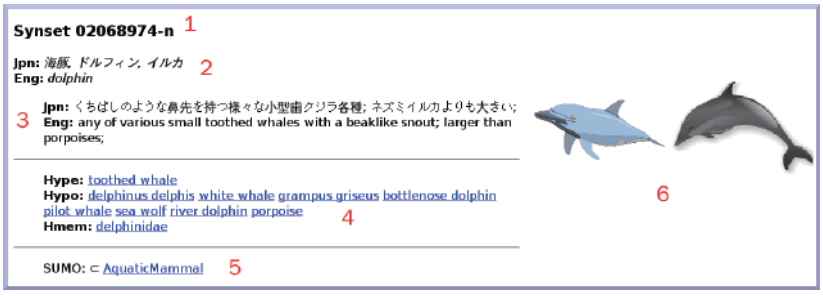
概念をどのように定義するのか

シンボル（文字列）に意味を与える方法がいくつかあります。

例えば、文法を利用する方法です。
　　クジラがプランクトンを食べた。
　　クジラ[名詞] が[助詞] プランクトン[名詞] を[助詞] 食べ[動詞] た[助動詞]。
この様に文法のパターンをいくつか用意すれば、それぞれの枠に入った文字列の意味を推定することができます。

概念をどのように定義するのか

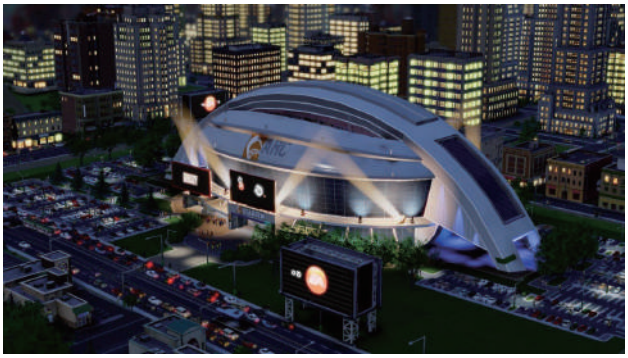
名詞動詞の関係性を整理した「概念辞書」というものを使用し、シンボル同士の関係性を定義することもできます。下の例だと、イルカの上位概念（Hype）は[toothed whale（歯鯨）]で、下位概念（Hypo）は[delphinus delphis（マイルカ）]などが定義されています。



出展（情報通信研究機構）：http://compling.hss.ntu.edu.sg/wnja/

現代の技術で何が解決できそうか

文法や概念辞書でシンボルの意味付けをする場合、文法から外れた言い方や概念辞書にない記号（スラング、新出の言葉など）を全て網羅することはできません。
一つの解決策として、現実世界を仮想空間に表現し、その中でコンピュータに状況を理解させるという考え方があるようです。
地理、建物、食べ物などを完璧に再現できれば、文法によらずに対象物や状況を表現できるようになるかも知れません。



出展（シムシティ）：https://www.ea.com/ja-jp/games/simcity/simcity

演習：シンボルグラウンディング問題

あなたは喫茶店で働くロボットだとします。
お客さんが2人が喫茶店に入ってきました。

客A「俺はコーヒー。砂糖もお願い。」
客B「じゃあ、アイスで。」

と客2人は注文しました。
この注文を理解するために、どのような情報、判断プロセスが必要になるでしょうか？

チューリングテスト

チューリングテストとは

[Wikipediaより]
チューリングテスト（英: Turing test）は、アラン・チューリングが提案した、ある機械が「人間的」かどうかを判定するためのテストである。これが「知的であるかかどうか」とか「人工知能であるかどうか」とかのテストであるかどうかは、「知的」あるいは「(人工)知能」の定義、あるいは、人間が知的であるか、人間の能力は知能であるか、といった定義に依存する。

チューリングテストは「機械が人に近い振る舞いができるかどうかを判別すること」であり、それが知性か否かを判断するもの、とはされていません。

知性とは

知性とは何かを考えさせられる実験として、「中国語の部屋」という実験があります。

[Wikipedia]より
ある小部屋の中に、アルファベットしか理解できない人を閉じこめておく（例えば英国人）。この小部屋には外部と紙きれのやりとりをするための小さい穴がひとつ空いており、この穴を通して英国人に1枚の紙きれが差し入れられる。そこには彼が見たこともない文字が並んでいる。これは漢字の並びなのだが、英国人の彼にしてみれば、それは「★△◎▽☆口」といった記号の羅列にしか見えない。 彼の仕事はこの記号の列に対して、新たな記号を書き加えてから、紙きれを外に返すことである。どういう記号の列に、どういう記号を付け加えればいいのか、それは部屋の中にある1冊のマニュアルの中に全て書かれている。例えば"「★△◎▽☆口」と書かれた紙片には「■@◎▽」と書き加えてから外に出せ"などと書かれている。
彼はこの作業をただひたすら繰り返す。外から記号の羅列された紙きれを受け取り（実は部屋の外ではこの紙きれを"質問"と呼んでいる）、それに新たな記号を付け加えて外に返す（こちらの方は"回答"と呼ばれている）。すると、部屋の外にいる人間は「この小部屋の中には中国語を理解している人がいる」と考える。しかしながら、小部屋の中には英国人がいるだけである。彼は全く漢字が読めず、作業の意味を全く理解しないまま、ただマニュアルどおりの作業を繰り返しているだけである。それでも部屋の外部から見ると、中国語による対話が成立している。

演習：知性とは

何ができたら知性とみなせるのか？
これは、そもそも「知性とはなにか？」という定義に関わる、非常に難しい問いです。

みなさんにとって、知性とは何でしょうか？

第3回：ルールベースの人工知能

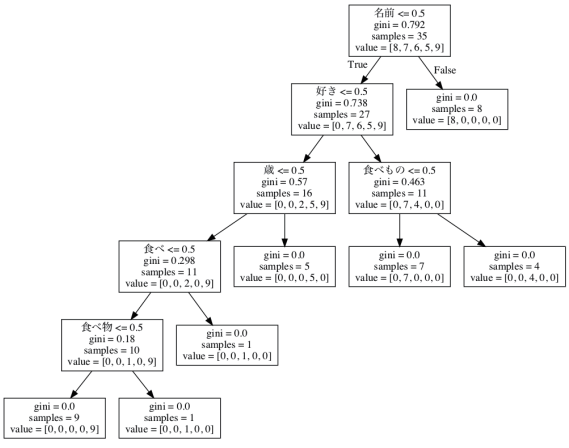
アジェンダ

- 決定木
- ルールベースの人工知能
- 知識ベースの人工知能
- エキスパートシステム

決定木

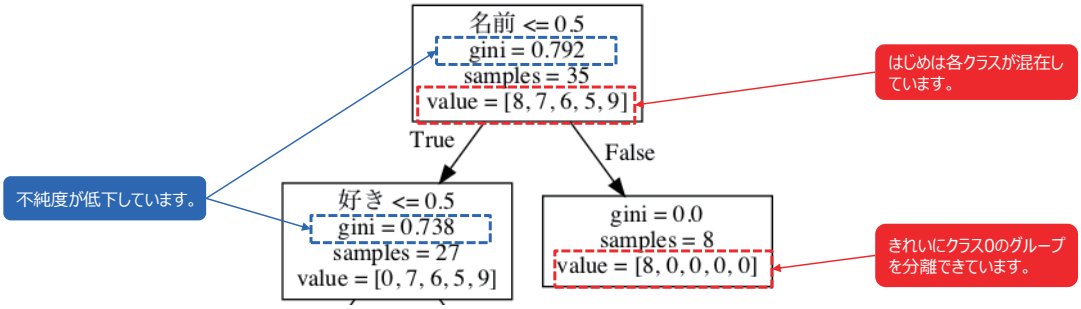
決定木とは

- 決定木分析は回帰や分類を実行する手法です。
- 顧客情報やアンケート結果などの目的変数に寄与する説明変数を見つけ、樹木状のモデルを作成します。



決定木の分岐の指標

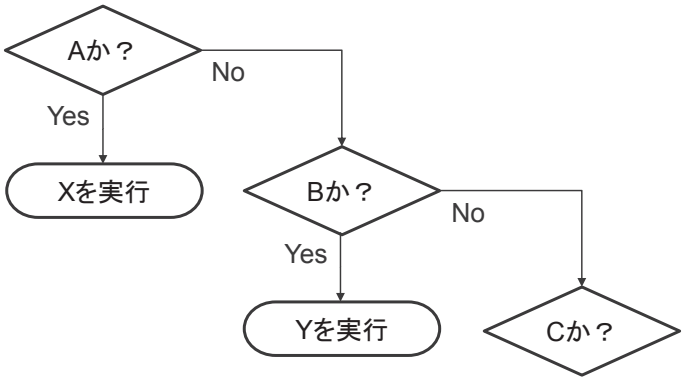
- 決定木においてデータを分割する基準は「情報利得」と「不純度」です。
- 情報利得とは、いかにうまく分割できたか（分割前の不純度 − 分割後の不純度）を表します。
- 不純度とは、分割後のグループにおける、複数のクラスの混在の度合い（どれだけごちゃごちゃしているか）を表します。



ルールベースの人工知能

ルールベースの人工知能

ルールベースとは、「どのような状況なら何を実行するのか」を明確に定義する方法です。
・もしAならXを実行する。AでないならYを実行する。
・もしAならXを実行する。BならYを実行する。AでもBでもなくCならZを実行する。



ルールをどのように構築するか

ルールベースシステムのアルゴリズムを構築するためには、システムが対象とする分野の知識が必要になります。

例えば「ホットプレートの温度管理」を考えます。
温度がある一定値（閾値）を超えたらスイッチを切って温度上昇を抑え、反対に閾値を下回ったらスイッチを入れ、温度を一定に保つ機能を実装する必要があるとします。

閾値はホットプレートの材質に依存するかも知れません。ホットプレートで何を作るか、に依存することもあるでしょう。
もしかしたら、安全に関わる法律も考慮する必要があるかも知れません。

このようにルールを構築するためには、対象となる分野の知識が必要になります。

ルールをどのように構築するか

MECE（ミーシー）という言葉があります。「漏れなくダブりなく」という意味で、ロジカルシンキングなどの分野でよく登場する言葉です。
ルールを作成するときは、「MECEになっているか？」を考える必要があります。

例えばホットプレートの温度設定で、2-A（平板で焼き肉を作る）を考慮し忘れていた場合、これは「漏れ」があるということになります。
もしくは、決定木において2つの枝で1-B（厚底鍋でお好み焼きを作る）が出てきたとします。これは「ダブリ」があることになります。

実際の業務において「漏れ」を防ぐことは困難ですので、ルールをすり抜けた「例外処理」を設けることが必要になります。
また、複雑なルールであるほど「ダブリ」が発生しやすくなります。
経路によって実行結果が変わらないようにルールを整備する必要があり、ルールのメンテナンスコストが肥大していきます。

		対象とする料理		
		1.お好み焼き	2.焼き肉	3.鍋
プレートの種類	A.平板	1-A	2-A	3-A
	B.厚底鍋	1-B	2-B	3-B

ルールをどのように構築するか

ルールの分岐（パラメータ）が増えるほど、ルールは複雑になっていきます。
[対象とする料理3つ]×[プレートの種類2つ]という単純な事例でも6パターンの挙動を考慮しなければなりません。
2-B（厚底鍋で焼き肉）はありえないだろう、ということで挙動を考慮しなくてよいのではなく、「2-Bは想定しないので挙動を考慮しない」という別のルールが必要になります。

このように、ルールベースシステムにおいてはパラメータ数が増えると、判定に要する時間が爆発的に増えてしまいます。

		対象とする料理		
		1.お好み焼き	2.焼き肉	3.鍋
プレートの種類	A.平板	1-A	2-A	3-A
	B.厚底鍋	1-B	2-B	3-B

演習：車のブレーキをルールベースで制御する

車の自動ブレーキ制御のプログラムを設計しているとします。
・車体の乾燥重量
・燃料の残量
・搭乗人数
・速度
・障害物までの距離
・ブレーキ踏み込み角度と制動力
など様々なパラメータを考慮する必要があると思われます。

これらのパラメータ（上記以外で必要と思われるものも含め）を、どのような順番で判定することが好ましでしょうか。

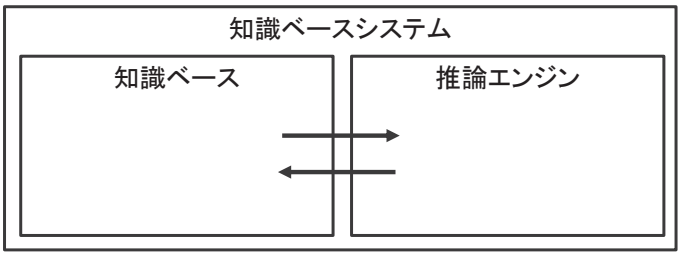
知識ベースの人工知能

知識ベースとは

[Wikipediaより]
知識ベース（knowledge base）はナレッジマネジメントのための特殊なデータベースであり、KBと略記されることもある。
それは知識の検索を可能とし、知識を組織化し、知識をコンピュータ上に集合させたものである。

知識ベースシステムとは

知識を利用して意思決定を支援するシステム、もしくは問題解決を支援するシステムを知識ベースシステムといいます。
知識ベースシステムは、知識を一定の形式で蓄積した知識ベースと、知識ベース内に蓄積された知識を活用して推論するための制御機構である推論エンジンで構成されます。



知識の表現

「～ならば、～である」というIF～THENで表現できる形で知識を蓄えていきます。
例えば、「肺炎か否か？」と判定するための知識として、以下のようなルールが考えられます。

ルールNo	IF	THEN	確信度
Rule1	くしゃみがでる or 鼻詰まりがある or 喉が痛む or 咳が出る	風邪の症状がある	1.0
Rule2	（熱が高い or 頭がいたい）and 風邪の症状がある	風邪である	1.0
Rule3	（関節が痛む or 胃の調子が悪い）and 風邪の症状がある	風邪である	0.8
Rule4	四日以上高熱が続いている and 風邪である	肺炎の疑いがある	0.5

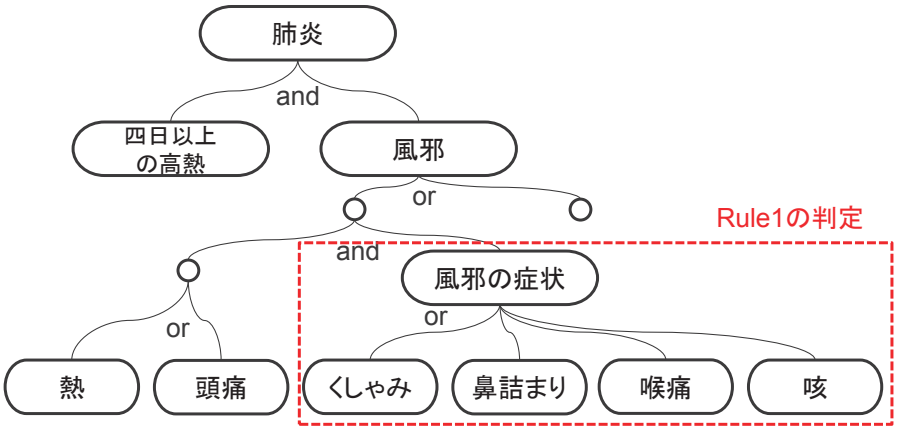
エキスパートシステム

エキスパートシステムとは

エキスパートシステムは、知識ベースシステムの一種です。
問題領域のエキスパートの知識を用いて推論を行い、高度で複雑な問題をエキスパートと同等レベルで解決できる
プログラムのことをエキスパートシステムといいます。

エキスパートシステムの例

例えば、肺炎か否かを判定するシステムがあるとします。
「知識の表現」のページに記載したルールの判定を行い、最終的に肺炎なのかを評価します。
ルールの判定の順番、確信度の取り扱いなどを推論エンジンが制御します。



エキスパートシステムと機械学習の共存

筆者は、「エキスパートシステムとして構築された金融機関のある不正取引検知システムを、機械学習を活用して高度化できないか？」という相談を受けたことがあります。

現行システムを調査し、また実データを用いてのPoC（概念実証）を実施した結果、「エキスパートシステムと機械学習のいいとこ取り」に落ち着くことになりました。

金融機関は規制当局の通達に従い規制をおこなうという、明確なルールベースに則る業務があります。また、現場のエキスパートの方が蓄積してきた知識ベースは、使用可能なデータのみですぐに機械学習で転用できるという類のものではありませんでした。

ただし、ある種のデータ群から不正利用を検知するという明確なタスクにおいて、機械学習の精度は圧倒的に人間を超えていました。

結局のところ、知識ベース（人間の知見と感の蓄積）でデータをグルーピングした後、それぞれのデータ群に対する不正検知については機械学習を適用する、というハイブリッド型の設計に落ち着きました。

演習：エキスパートシステムと機械学習

エキスパートシステムと機械学習が組み合わさった事例を調査してください。
どのような処理にエキスパートシステムを適用し、どのような処理に機械学習を適用しているのか整理比較してください。

第4回：画像処理

アジェンダ

- 画像処理の目的
- OpenCVによる画像処理
- OCR（光学文字認識）
- 物体認識

画像処理の目的

画像処理の目的

画像は「jpeg」や「png」などの拡張性で保存されます。
これらの「画像」は、そのままでは統計処理を実施したり機械学習器に投入することはできません。

各種アルゴリズムが扱いやすいように、画像を数値に変換する必要があります。
画像処理の目的の一つが、このように画像を数値に変換することです。

データの前処理とは

- 「学習」ステップにおいて機械学習などを活用してモデルを作成しますが、その前ステップとして「前処理」が必要となります。
- 前処理は、機械学習器を使用するためには必須である前処理と、モデル精度を向上させるために実施することが望ましい前処理とに大別できます。

モデル作成のステップ

必須の前処理とは

- 機械学習器は数値を扱うことができます。区分値や自然言語、画像などは機械学習器で扱える数値データに変換する必要があります。
- 例え数値データを扱うとしても、欠損値が存在する場合はそのままでは機械学習器に掛ける事ができない場合があります。

画像処理に用いられるライブラリの例

画像処理を実行できるライブラリの例を以下に示します。

OpenCV（C/C++、Java、Python）
PIL / pillow（Python）
skimage（Python）

本講義では、主にOpenCVの事例について学習します

openCVとは

openCVとは

openCV（出展：https://ja.wikipedia.org/wiki/OpenCV）

- OpenCV（オープンシーヴィ、英語: Open Source Computer Vision Library）とはインテルが開発・公開したオープンソースのコンピュータビジョン向けライブラリ。2009年にWillow Garage（ウィロー・ガレージ）に開発が移管された後、2015年現在はItseezがメンテナンスを行なっている。なお、2016年5月26日にインテルがItseezを買収することが発表された。

コンピュータビジョン（出展：https://ja.wikipedia.org/wiki/コンピュータビジョン）

- コンピュータビジョン（computer vision）はコンピュータがデジタルな画像、または動画をいかによく理解できるか、ということを扱う研究分野である。工学的には、人間の視覚システムが行うことができるタスクを自動化することを追求する分野である。

openCVとは

OpenCVは下記のような関数を搭載しており、ビジョンの研究者や開発者の仕事を楽にするためのツールです。

- 線形代数や統計処理など，コンピュータビジョンに必要な各種数学関数
- 直線や曲線，テキストなど画像への描画関数
- OpenCVで使用したデータを読み込み/保存するための関数
- エッジ等の特徴抽出や画像の幾何変換，カラー処理等々の画像処理関数
- 物体追跡や動き推定などの動画像処理用関数
- 物体検出などのパターン認識関数
- 三次元復元のためのカメラ位置や姿勢の検出などのカメラキャリブレーション関数
- コンピュータにパターンを学習させるための機械学習関数
- 画像の読み込みや保存，表示，ビデオ入出力などインターフェース用関数

openCVで実施できる処理

画像データの読み込み

- 画像を読み込むと3次元配列がピクセル分連なった行列に変換されます。
- 3次元配列の中身はBGRの順番となっています。

```
# 読み込みたい画像のパスを指定します。
path_img = 'resources/images/books.jpg'

"""
画像を読み込みます。読み込むと数値に変換されます。
3次元配列(BGR)がピクセル数連なった行列に変換されます。
"""
img = cv2.imread(path_img)

# 行列の中身を確認します。
print(img)
```

出展：https://github.com/piratefsh/image-processing-101/tree/master/images

```
[[[160 157 153]
  [158 155 150]
  [153 151 143]
  ...
  [133 161 162]
  [129 160 163]
  [129 163 163]]

 [[158 157 153]
  [155 155 149]
  [150 151 142]
  ...
```

色の要素の順番変更

- OpenCVでは画像はBGRフォーマットで読み込まれる初期設定となっていますが、Matplotlibの場合はRGBで読み込まれます。R:赤とB:青の数値が入れ替わっているまま表示させると、下図のように色調が異なっていることが確認できます。

出展：https://github.com/piratefsh/image-processing-101/tree/master/images

matplotlibによって「BGR」で表示

matplotlibによって「RGB」で表示

グレースケール化

- グレースケール画像とは、最も暗い色（黒）を表す数値を0、また最も明るい色（白）を表す数値を255とし、ピクセルの明るさを表現する256階調のスケールで、1つの色チャンネルしか持っていない画像のことです。

出展：https://github.com/piratefsh/image-processing-101/tree/master/images

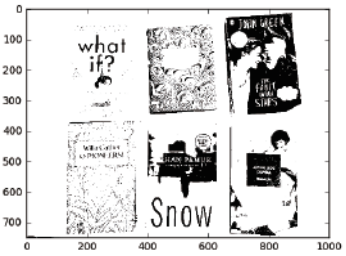
グレースケール化した画像

閾値処理

- ピクセルのチャンネル値がある閾値を超えた場合は、画像の各ピクセルを白いピクセルに、超えなかった場合は、黒いピクセルに置き換えます。
- 画像を2値画像、つまりシングルチャンネル画像に変換する処理で、画像セグメンテーションの一番簡単な方法です。
- グレースケール化はシングルチャンネル画像の一種です。

出展：https://github.com/piratefsh/image-processing-101/tree/master/images

グレースケール化した画像



閾値処理を実施した画像

画像のフィルタリング処理：畳み込み

- 学習データやテストデータにノイズ除去や画像のぼかしを入れると、汎化性能の高いロバスト（頑健な）なモデル作成や判定を行うことができます。
- openCVでは入力画像とカーネル(フィルタ)のconvolution（畳み込み）を計算できます。以下に5x5サイズの平均値フィルタに使うカーネルを示します。

$$K = \frac{1}{25}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

出展：http://labs.eecs.tottori-u.ac.jp/sd/Member/oyamada/OpenCV/html/py_tutorials/py_imgproc/py_filtering/py_filtering.html#filtering

画像のフィルタリング処理：ガウシアンフィルタ

ガウシアンフィルタ（出展：http://labs.eecs.tottori-u.ac.jp/sd/Member/oyamada/OpenCV/html/py_tutorials/py_imgproc/py_filtering/py_filtering.html#filtering）

- 注目画素との距離に応じて重みを変えるガウシアンカーネルをフィルタに使用する事もできます。
- カーネルの縦幅と横幅(どちらも奇数)に加え、ガウシアンの標準偏差値sigmaX(横方向)とsigmaY(縦方向)を指定する必要があります。sigmaXしか指定されなければ、sigmaYはsigmaXと同じだとみなされます。どちらの値も0にした場合、カーネルのサイズから自動的に計算されます。
- ガウシアンフィルタは白色雑音の除去に適しています。

OCR (Optical character recognition)

OCRとは

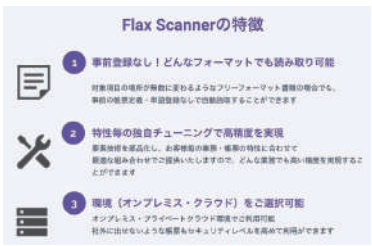
[Wikipediaより]
光学文字認識（こうがくもじにんしき、Optical character recognition）は、活字の文書の画像（通常イメージスキャナーで取り込まれる）を文字コードの列に変換するソフトウェアである。一般にOCRと略記される。OCRは、人工知能やマシンビジョンの研究分野として始まった。研究は続けられているが、OCRの中心はその実装と応用に移っている。紙に印刷された文書をデジタイズし、よりコンパクトな形で記録するのに必要とされる。さらに、文字コードに変換することで機械翻訳や音声合成の入力にも使えるようになり、テキストマイニングも可能となる。研究分野としては、パターン認識、人工知能、コンピュータビジョンが対応する。

AI-OCRの事例

近年、OCRに機械学習を応用したプロダクト、サービスを展開する企業が搭乗してきました。
定形フォーマットにおける読み取り精度の向上に注力する企業、非定形フォーマットでの読み取りに注力する企業など、各社独自の強みを発揮してAI-OCR市場を牽引しているようです。

出展（AI inside 株式会社）：https://inside.ai/

出展（株式会社 Cogent Labs）：https://www.tegaki.ai/



出展（株式会社シナモン）：https://cinnamon.is/flaxscanner

物体認識

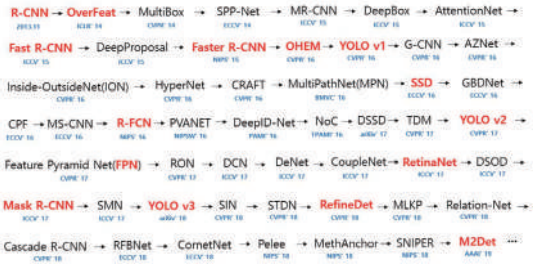
物体認識とは

物体認識は一般物体認識と特定物体認識に大別されます。

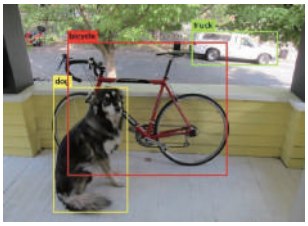
特定の対象物のみを認識するタスクは特定物体認識と呼ばれます。
一方、様々な対象物をまとめて認識するタスクは、一般物体認識と呼ばれます。

一般物体認識

画像中の物体の位置を検出し、かつ物体の名前を予測することです。
世界中で様々なアルゴリズムが開発されています。



出展：https://github.com/hoya012/deep_learning_object_detection

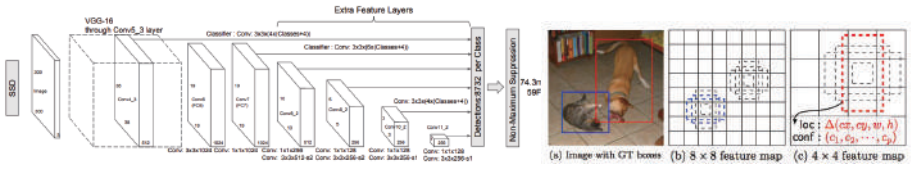


出展：https://pjreddie.com/darknet/yolo/

一般物体認識器の構築

一般物体認識について学習したい方は、例えば以下の論文をご参照ください。
https://arxiv.org/abs/1512.02325

判定領域のサイズを変えながらクラス分類をするタスクを繰り返し、モデルの重みを最適化します。



第5回：パターン検索

アジェンダ

- パターン検索
- 正規表現を使用したパターン検索
- 高度なパターン検索

パターン検索

パターン検索とは

何かしらのデータを検索するときに、データ出現のパターンを利用して検索することをパターン検索といいます。
例えば、文字列のパターン検索には、以下のようなものがあります。

- 文字列一致
- 前方一致
- 後方一致

また、正規表現といわれる記述を使用し、以下のようなパターン検索もあります。

- 任意文字指定
- 部分文字指定
- 範囲指定

パターン検索：文字列一致

検索対象文字列と指定したキーワードが一致するかどうかを調べます。

（例）
検索対象文字列：
天気
検索文字列：
天気
検索結果：
True（検索文字列が、検索対象文字列と一致した。）

パターン検索：前方一致

検索対象文字列の先頭に、指定したキーワードが存在するかどうかを調べます。

（例）
検索対象文字列：
天気がよい
検索文字列：
天気
検索結果：
True（検索文字列が、検索対象文字列の先頭に存在した。）

パターン検索：後方一致

検索対象文字列の末尾に、指定したキーワードが存在するかどうかを調べます。

（例）
検索対象文字列：
天気が悪い一日
検索文字列：
一日
検索結果：
True（検索文字列が、検索対象文字列の末尾に存在した。）

正規表現を使用したパターン検索

正規表現とは

※本講義では、主にpythonを使用することを前提として学習していきます。

正規表現とは、文字列の集まりを1つの形式で表すための特別な書き方です。
例えば、以下のような正規表現があります。

メタ文字	説明	使用例	合致する文字列	合致しない文字列
.	任意の1文字	h.t	hat,hot,hit	head,height
*	直前のパターンを0回以上繰り返す	山川*	山,山川,山川川	里
+	直前のパターンを１回以上繰り返す	gre*n	greeeeeeen	grn
?	直前のパターンを0もしくは1回繰り返す	山川?	山,山川	山川川
{数字}	直前のパターンを数字の分だけ繰り返し	9{3}	999	9,99,9999
[文字,文字]	カッコ内の文字のいずれか１文字	[a,b,c]	a,b,c	d,e

特殊シーケンス	説明	例
\d	任意の数字	1,2,3など
\D	任意の数字以外	a,b,abcなど
\w	任意の英数字	1,2,333,ab,12abなど
\W	任意の英数字以外	山、川、空白など

参照：https://www.sejuku.net/blog/23232

パターン検索：任意文字指定

検索対象文字列に、任意の文字が入ったキーワードが含まれているかを調べます。

```
# 必要なライブラリをインポートします。
import re

# 検索する文字列です。猫山グループを検索します。
pattern = "猫山.*株式会社"

# 検索対象の文字列です。
text = "猫山株式会社"

# textの中からpatternに一致する文字を抽出します。
matchedList = re.findall(pattern, text)
if matchedList:
    print(matchedList)
else:
    print("検索結果なし")
```

['猫山株式会社']

```
# 検索対象の文字列です。
text = "猫山システムズ株式会社"

# textの中からpatternに一致する文字を抽出します。
matchedList = re.findall(pattern, text)
if matchedList:
    print(matchedList)
else:
    print("検索結果なし")
```

['猫山システムズ株式会社']

←

「猫山」と「株式会社」の間に「.*」という正規表現を記載しています。
「.」は任意の一文字という意味です。
「*」は直前のパターンを0回以上繰り返すという意味です。
「.*」と2つの正規表現を連続させることによって、任意の一文字が0回以上繰り返す、という意味になります。

よって「猫山」と「株式会社」の間に任意の文字が存在する（存在しない場合も含めて）パターンを抽出することができます。

パターン検索：部分文字指定

指定したキーワードの間に特定の文字列が入っているパターンを検索します。

```
# 検索する文字列です。
# 猫山グループのうち、システム会社とコンサルティング会社のみを検索します。
pattern = "猫山(システムズ|コンサルティング)株式会社"
```

「猫山」と「株式会社」の間に、「()」で囲んで「システムズ」と「コンサルティング」いう文字列を記載しています。
また、パイプと呼ばれる「|」はor（もしくは）の意味です。

「猫山」と「株式会社」の間に「システムズ」もしくは「コンサルティング」が存在する場合のみ検索することができます。

```
# 検索対象の文字列です。
text = "猫山システムズ株式会社"

# textの中からpatternに一致する文字を抽出します。
matchedList = re.findall(pattern, text)
if matchedList:
    print(matchedList)
else:
    print("検索結果なし")
```

```
['システムズ']
```

```
# 検索対象の文字列です。
text = "猫山コンサルティング株式会社"

# textの中からpatternに一致する文字を抽出します。
matchedList = re.findall(pattern, text)
if matchedList:
    print(matchedList)
else:
    print("検索結果なし")
```

```
['コンサルティング']
```

パターン検索：範囲指定

指定したキーワードの間に範囲指定した文字列が入っているパターンを検索します。

```
# 検索対象の文字列です。
text = "今日は朝の9時から11時まで散歩しました。歩いた距離は7kmでした。"
```

```
# 検索する文字列です。数値を取り出します。
#pattern = "\d"
# こちらの表現でも同様の動きをします。
pattern = "[0-9]"
```

「[0-9]」と記載すると、数字のみを検索することができます。
「＼d」でも同様の動きをします。

```
# コンパイルします。
re_compiled = re.compile(pattern)
```

```
# textの中からpatternに一致する文字を抽出します。
matchedList = re_compiled.findall(text)
if matchedList:
    print(matchedList)
```

```
['9', '1', '1', '7']
```

高度なパターン検索

正規表現の組み合わせ

例えば、電話番号をパターンマッチングで取得することを考えます。
[https://sites.google.com/site/diveintopythonjp/home/7-seiki-hyougen/7-6-kesu-sutadi--denwa-bangou-no-koubun-kaiseki]

電話番号の記載のパターンを観察することで、正規表現を
複数組み合わせて検索することができます。
参考になるソースはwebですぐに見つけることができますので、
興味のある方は検索してみてください。

- 800-555-1212
- 800 555 1212
- 800.555.1212
- (800) 555-1212
- 1-800-555-1212
- 800-555-1212-1234
- 800-555-1212x1234
- 800-555-1212 ext. 1234
- work 1-(800) 555.1212 #1234

Example 7.10. 番号を見つけ出す

```
>>> phonePattern = re.compile(r'^(\d{3})-(\d{3})-(\d{4})$')  ❶
>>> phonePattern.search('800-555-1212').groups()  ❷
('800', '555', '1212')
>>> phonePattern.search('800-555-1212-1234')  ❸
>>>
```

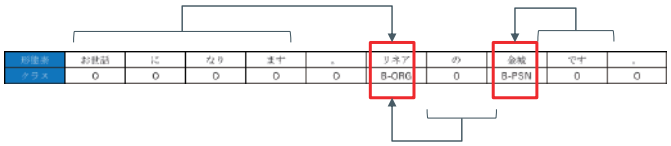
「意味」から文字列を取得する

言語の文法にしたがって文字列の出現パターンを解析するという方法もあります。
興味のある方は検索してみてください。
[https://qiita.com/Hironsan/items/326b66711eb4196aa9d4]

まず形態素解析を実施します。

```
(anaconda3-2.4.1) LINEAnoMacBook-puro:~ linea$ mecab
お世話になります。リネアの金城です。
お世話	名詞,サ変接続,*,*,*,*,お世話,オセワ,オセワ
に	助詞,格助詞,一般,*,*,*,に,ニ,ニ
なり	動詞,自立,*,*,五段・ラ行,連用形,なる,ナリ,ナリ
ます	助動詞,*,*,*,特殊・マス,基本形,ます,マス,マス
。	記号,句点,*,*,*,*,。,。,。
リネア	名詞,一般,*,*,*,*,*
の	助詞,連体化,*,*,*,*,の,ノ,ノ
金城	名詞,一般,*,*,*,*,金城,キンジョウ,キンジョー
です	助動詞,*,*,*,特殊・デス,基本形,です,デス,デス
。	記号,句点,*,*,*,*,。,。,。
EOS
```

CRF（条件付き確率場）で会社名と氏名を抽出します。



クラス	例
ART 固有物名	ノーベル文学賞、Windows7
LOC 地名	アメリカ、千葉県
ORG 組織	自民党、NHK
PSN 人名	安倍晋三、メルケル
DAT 日付	1月29日、2016/01/29
TIM 時間	午後三時、10:30
MNY 金額	241円、8ドル
PNT 割合	10%、3割

演習

演習1：任意文字指定によるパターン認識

- 指定したキーワードの間に任意の文字列が入っているパターンの検索を実施してください。
- パターンに合致しない場合に検索できないことも確認してください。

演習2：部分文字指定によるパターン認識

- 指定したキーワードの間に特定の文字列が入っているパターンの検索を実施してください。
- パターンに合致しない場合に検索できないことも確認してください。

演習3：範囲指定によるパターン認識

- 指定したキーワードの間に数値が入っているパターンの検索を実施してください。
- パターンに合致しない場合に検索できないことも確認してください。

第6回：人工知能概論総復習その1

第1回：人工知能を取り巻く状況

人工知能の歴史

人工知能学会の記事[https://www.ai-gakkai.or.jp/whatsai/AIhistory.html]を参考にしています。

この記事の中では、人工知能の歴史を5つに分けて記述しています。

1. 人工知能の夜明け（～1956）
2. 古き良き人工知能（1957～1969）
3. 現実からの反撃（1970～1979）
4. 人工知能の産業化（1980～1988）
5. 現在そして未来の彼方へ（1989～）

人工知能の夜明け（～1956）

1943 W.McCulloch（ウォーレン・マカロック）とW.Pitts（ウォルター・ピッツ）が“A Logical Calculus of the Ideas Immanent in Nervous Activity”を出版．ニューラルネットワークの基礎となりました．

1943 A.Rosenblueth（ローゼンブリュート），N.Wiener（ウィーナー ），J.Bigelow（ビゲロー）が論文で“サイバネティクス”という言葉を用いました．

- サイバネティクスとは、機械の自動制御や動物の神経系機能の類似性や関連性をテーマに研究する、心理学、生物学、物理学、数学等を包括した科学の総称です。
- アメリカの情報理論の大家であるノーバート・ウィーナー（Norbert Wiener）が、1948年に初めて発表したサイバネティクス理論は、生物と機械の間に情報のやりとりやコントロールの仕組みなどに関する類似性があることに着目し、自然から人工機構まで含めた多種多様な学問領域が協同して取り組む新しい研究課題への道を切り開きました。

人工知能の夜明け（～1956）

1950 A.M.Turingが“Computing Machiery and Intelligence”を出版．知的活動をテストする方法としてチューリングテストを示しました．

- 「機械が思考したかどうかは、人との会話が成立したかどうかで判断する。」と定義したテストです。人の審査員が、相手が見えない状況で「人」もしくは「コンピュータ」と対話し、相手が人なのかプログラムなのかを言い当てました。審査員がプログラムと対話した後に「『人と』対話した」と間違った答えを出せば、「機械は思考した」ということにしました。

1950 I.Asimov がロボット3原則を発表しました．「人間を傷つけてはななならい．傷つくのを看過してはならない」「第1原則に反しない限り，人間の命令に従わなくてはならない」「第1，第2原則に反しない限り自分の身を守らなくてはならない」

古き良き人工知能（1957～1969）

1957 J.Backus（ジョン・バッカス）が最初の高級言語FORTRANを開発．

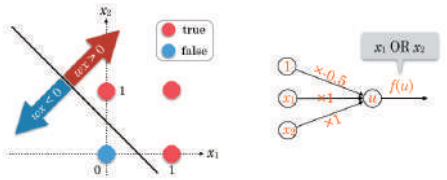
1965 J.Weizenbaum（ジョセフ・ワイゼンバウム）がELIZA（イライザ）を開発．英語でいろいろな話題について会話ができるプログラムで，精神科医をまねたバージョンはネットワーク上で人気を集めた．

- イライザは「考えて回答」しているわけではなく、会話のパターンを利用していました。例えば人がイライザに「お腹が痛い」と言えば、イライザは「なぜお腹が痛いのか？」と返すようになっていました。仕込んだパターン以外の質問をイライザにすると、イライザは回答できませんでした。

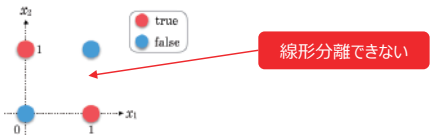
古き良き人工知能（1957～1969）

1968 M.MinskyとS.PapertがPerceptronsを出版し単層ニューラルネットであるパーセプトロンの限界を指摘.

単純パーセプトロンでは線形非分離な問題が解決できない。



OR（論理和）の例



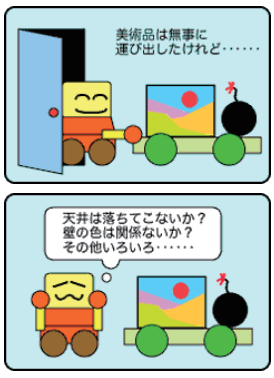
NAND（否定論理積）の例

出展： http://hokuts.com/2015/12/04/ml3-mlp/

古き良き人工知能（1957～1969）

1969 J.McCarthyとP.J.Hayesが人工知能最大の難問“フレーム問題”を指摘.

[人工知能学会HPより]
フレーム問題は，今からしようとしていることに関係のあることがらだけを選び出すことが，実は非常に難しいという問題です.
人工知能搭載のロボット「安全くん1号」は，人間の代わりに危険な作業をするロボットです．爆弾が仕掛けられている部屋から貴重な美術品を取り出してこなければなりません．安全くん1号は美術品の入った台車を押して美術品をとってきましたが，不幸なことに爆弾は台車にしかけられていたので，安全くんは爆発に巻き込まれてしまいました.
これは安全くん1号が，美術品を取り出すために荷車を押せばよいということは分かったのですが，そのことによって，爆弾も一緒に取り出してしまうということは分からなかったためでした.

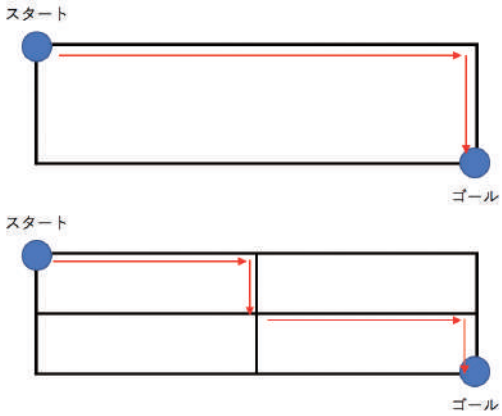


出展（人工知能学会）：http://hokuts.com/2015/12/04/ml3-mlp/

現実からの反撃（1970～1979）

1973 ライトヒル勧告．組合せ爆発問題を指摘した勧告．イギリスで2大学を除きAI研究の補助金がうち切られる．

「格子状の道をスタート地点からゴールまで同じところを2度と通らない道順の数」は、格子の数が増えると爆発的に増えます。



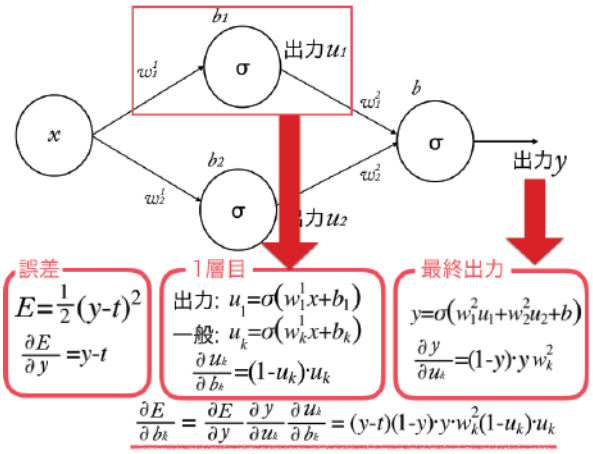
人工知能の産業化（1980～1988）

1982 日本で第5世代プロジェクトの開始．超並列で論理型言語を実行するコンピュータと自然言語の理解などを目標としました．

[Wikipediaより]
1980年代に入り、日本のコンピュータ産業は輸出も増え、市場規模も2兆円まで成長した。従来、通産省は1983年ごろまで貿易自由化対策としてコンピュータ企業への助成金を出していたが、既にそのような直接的な助成金は意義を失っていた。また、海外からもIBM互換機を輸出する日本に対して風当たりが強くなっていた時期でもある（IBM産業スパイ事件が起きたのは1982年）。そこで、次は第四世代と言われていた時代に、あえて更に先の第五世代コンピュータを開発するプロジェクトを立ち上げ、日本の独自性を打ち出そうとした。
この検討が開始されたのが1979年である。当時、電子技術総合研究所（現在の産業技術総合研究所）の渕一博らは述語論理によるプログラミングに強い関心を持っていた。渕らは独創性を求めるこのプロジェクトを絶好の機会として働きかけ、第五世代コンピュータの目標は「述語論理による推論を高速実行する並列推論マシンとそのオペレーティングシステムを構築する」というものになった。
当初の予定から1年延びた1992年、プロジェクトは「当初の目標を達成した」として完了した。

人工知能の産業化（1980～1988）

1980中 ニューラルネットのバックプロパゲーション・アルゴリズムが広く用いられるようになる.

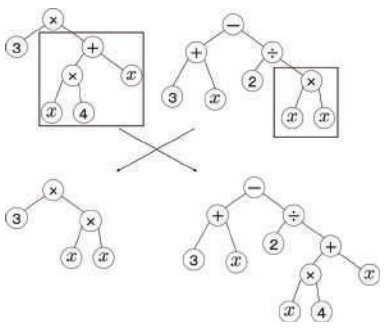


出展： https://www.yukisako.xyz/entry/backpropagation

現在そして未来の彼方へ（1989～）

1990 J.R.Kozaが遺伝的プログラミングを開始.

[Wikipediaより]
遺伝的プログラミングは1990年にジョン・コザ（John Koza）によって提案された。他の進化的アルゴリズムの主要な方法論が同時期に提案され独立して研究が進められていたのに対し、遺伝的プログラミングは最初から遺伝的アルゴリズムの拡張として提案されており、他の三つの方法とは大きく立場を異にする。



現在そして未来の彼方へ（1989～）

1990中 データマイニング技術の誕生.

- データマイニング（Data mining）とは、統計学、パターン認識、人工知能等のデータ解析の技法を大量のデータに網羅的に適用することで知識を取り出す技術・プロセスの総称です。
- 英語では“Data mining”の語の直接の起源となった研究分野であるknowledge-discovery in databases（データベースからの知識発見）の頭文字をとってKDDとも呼ばれます。

人事分野への応用

- 自然言語処理により、応募者が記載したエントリーシートの内容を評価する事例です。
- 人事担当者の負担の軽減、評価基準の統一などの効果があります。

プレスリリース 2017年

いいね！ ツイート　一覧へ戻る | < 前のリリース | 次のリリース >

新卒採用選考におけるIBM Watsonの活用について

2017年5月29日
ソフトパンク株式会社

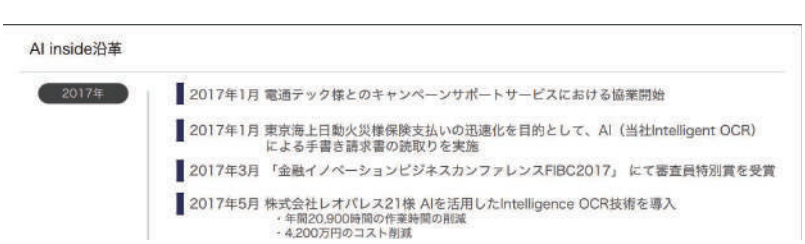
ソフトバンク株式会社は、応募者をより客観的に、また適正に評価することを目的に、2017年5月29日より新卒採用選考のエントリーシート※評価にIBM Watson日本語版（以下「IBM Watson」）を活用します。

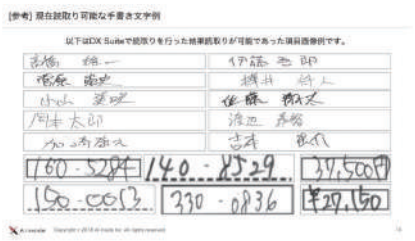
過去のデータを学習させたIBM Watsonに応募者のエントリーシートデータを読み込ませると、IBM WatsonのAPIの一つであるNLC（Natural Language Classifier、自然言語分類）により、エントリーシートの内容が認識され、項目ごとに評価が提示されます。合格基準を満たす評価が提示された項目については、選考通過とし、それ以外の項目については人事担当者が内容を確認し、合否の最終判断を行います。IBM Watsonによる評価をエントリーシート選考の合否判断に活用することで、統一された評価軸でのより公平な選考を目指します。

出展：https://www.softbank.jp/corp/group/sbm/news/press/2017/20170529_01/

業務効率化

- OCR（Optical Character Recognition：光学的文字認識）により手書き文字をAIが認識する事例です。
- 事務作業の効率化ができます。





出展：https://rpa-bank.com/pdf/document-dl/ai_inside.pdf

新商品開発

- 保険会社が新商品開発にAIを使用した事例です。
- 保険会社は新しい商品の売上が増え、保険を契約する顧客は機器の保守コストを抑えることができる事例です。

ニュース

東京海上日動と日立が保険サービスを共同開発、IoTとAIで予兆診断

山端 宏実＝日経 xTECH／日経コンピュータ　日経 XTECH

この記事を評価する

この記事は ○仕事に役立った0 ○人に勧めたい0 ○難しい0 ○易しい0

東京海上日動火災保険と日立製作所は2019年1月16日、データを活用した保険サービスを共同開発することで合意したと発表した。東京海上日動が持つ事故データや保険サービスと、日立のIoT（インターネット・オブ・シングズ）やAI（人工知能）などの技術を組み合わせ、新たな保険サービスを生み出す狙いだ。

第1弾として月内に、日立のIoTやAIによる予兆診断技術を活用し、製造設備の異常の兆候をつかみ、検査などに必要な費用を補償する保険の提供を始める。従来の保険は物的損壊を補償の要件にしていた。設備の異常の兆候を検知し、あらかじめ対策を講じることができれば企業は損失を最小限に抑えられ、保険会社が支払う補償額も少なく済む利点がある。

出展：https://tech.nikkeibp.co.jp/atcl/nxt/news/18/03854/

販促活動

- アサヒビールが、自社製品を販売してくれる小売企業（スーパー）に対して値付けサービスを提供するという事例です。
- アサヒビールを取り扱う小売企業が潤えば、アサヒビールも潤うというサプライチェーン全体を見据えた取り組みの事例です。

アサヒ、AIでビール売価指南　販売てこ入れ

2018/8/21 20:58 | 日本経済新聞　電子版

保存 共有 印刷 その他

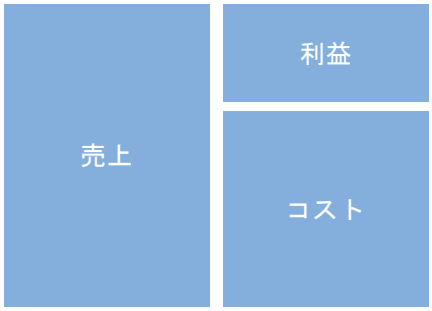
アサヒビールは、スーパーなどがビール系飲料の収益を効率良く上げるため、最適な値付けを提案できるシステムを売り込む。人工知能（AI）が販売環境などを踏まえ予測する。2017年6月の酒の安売り規制強化後、販売競争は激しい。小売店はライバル店の動きを横目に原価を下回らずに値ごろ感をどう出すか悩んでいる。競争力のある価格設定を手助けし、小売りとの取引拡大につなげる。

システムはNECのAI技術を組み込み、同社と共同開発した。小売店からPOS（販売時点情報管理）データに加え、天気や気温、運動会といった周辺イベント情報の提供を受けAIに入力する。入力データをもとに小売店の売り上げや利益を効率良くあげるにはどんな商品を、どの時期にどんな価格で売ればよいかAIがはじき出す。

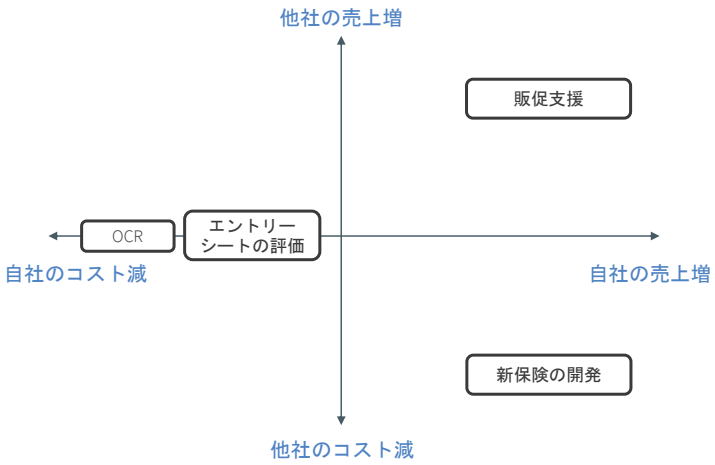
出展：https://www.nikkei.com/article/DGXMZO34407380210820189l6M00/

AIをどこで使うか

- 企業活動においてAIを適用する効果として①売上が増えるのか、もしくは②コストが減るのかのどちらかに大別できます。



企業における収支のバランス



第2回：人工知能の各種問題

フレーム問題と一般化フレーム問題

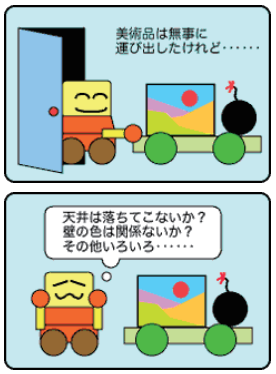
[Wikipediaより]
フレーム問題（フレームもんだい、英: Frame problem）とは、人工知能における重要な難問の一つで、有限の情報処理能力しかないロボットには、現実に起こりうる問題全てに対処することができないことを示すものである。1969年、ジョン・マッカーシーとパトリック・ヘイズ（英語版）の論文[1]の中で述べられたのが最初で、現在では、数多くの定式化がある。

現実世界で人工知能が、たとえば「マクドナルドでハンバーガーを買え」のような問題を解くことを要求されたとする。現実世界では無数の出来事が起きる可能性があるが、そのほとんどは当面の問題と関係ない。人工知能は起こりうる出来事の中から、「マクドナルドのハンバーガーを買う」に関連することだけを振るい分けて抽出し、それ以外の事柄に関して当面無視して思考しなければならない。全てを考慮すると無限の時間がかかってしまうからである。つまり、枠（フレーム）を作って、その枠の中だけで思考する。
だが、一つの可能性が当面の問題と関係するかどうかをどれだけ高速のコンピュータで評価しても、振るい分けをしなければならない可能性が無数にあるため、抽出する段階で無限の時間がかかってしまう。
これがフレーム問題である。
あらかじめフレームを複数定義しておき、状況に応じて適切なフレームを選択して使えば解決できるように思えるが、どのフレームを現在の状況に適用すべきか評価する時点で同じ問題が発生する。

フレーム問題と一般化フレーム問題

1969 J.McCarthyとP.J.Hayesが人工知能最大の難問“フレーム問題”を指摘.

[人工知能学会HPより]
フレーム問題は，今からしようとしていることに関係のあることがらだけを選び出すことが，実は非常に難しいという問題です．
人工知能搭載のロボット「安全くん1号」は，人間の代わりに危険な作業をするロボットです．爆弾が仕掛けられている部屋から貴重な美術品を取り出してこなければなりません．安全くん1号は美術品の入った台車を押して美術品をとってきましたが，不幸なことに爆弾は台車にしかけられていたので，安全くんは爆発に巻き込まれてしまいました．
これは安全くん1号が，美術品を取り出すために荷車を押せばよいということは分かったのですが，そのことによって，爆弾も一緒に取り出してしまうということは分からなかったためでした．



出展（人工知能学会）：http://hokuts.com/2015/12/04/ml3-mlp/

人間が自然に実現しているフレームの例

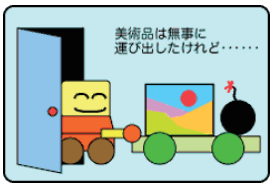
日常生活において、人間は滅多にフレーム問題に悩まされません。
これは人間が有限サイズのフレームで情報を囲っているからだと考えられます。つまり、無限の情報を処理する必要はなく、そのフレーム内の情報のみを処理すればよい、ということです。

人間がフレーム問題をうまく処理している例を挙げます。

- 「いつものところに6時」
  当事者間に共通の認識がある場合、これほど曖昧でも約束は成立します。
  つまり情報がフレームで囲われているため、指示対象が一つに決定しています。
- 「他人に迷惑をかけるな」
  当事者同士がどのような状況に置かれているのか、何を指して発言しているのか共通認識がある場合、これが成立します。
  フレームで情報を囲わないと、考慮するべき対象が無限に広がってしまい、処理能力が足りずにパンクします。

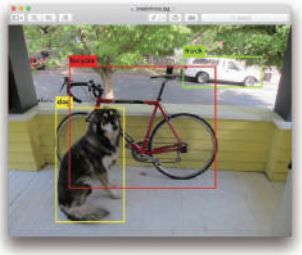
現代の技術で何が解決できそうか

[人工知能学会HPより]
安全くん1号は美術品の入った台車を押して美術品を
とってきましたが，不幸なことに爆弾は台車にしかけられていたので，安全くんは
爆発に巻き込まれてしまいました.



出展（人工知能学会）：
http://hokuts.com/2015/12/04/ml3-mlp/

このような問題は、例えば画像認識（一般物体認識）と[if else]のロジックで解決できる部分が多々あると思われます。
まず、一般物体認識で美術品とそれ以外の物体を判別します。
次に、「美術品以外の物体を認識した場合は、それを排除する」というif else ロジックを実行します。



出展（YOLO）：
https://pjreddie.com/darknet/yolo/

演習：フレーム問題

フレーム問題に対して、現在の技術を使って何が解決できそうか、議論してください。

シンボルグラウンディング問題とは

[Wikipediaより]
シンボルグラウンディング問題（シンボルグラウンディングもんだい）とは、記号システム内のシンボルがどのようにして実世界の意味と結びつけられるかという問題。記号接地問題とも言う。ハルナッド(w:Stevan Harnad)によって命名された。

例えば「雪」を一度も見たことがない南国の人は、「雪」という記号（シンボル：文字列）を与えられただけでは「雪」が一体何なのかを理解できません（グラウンディングできない）。

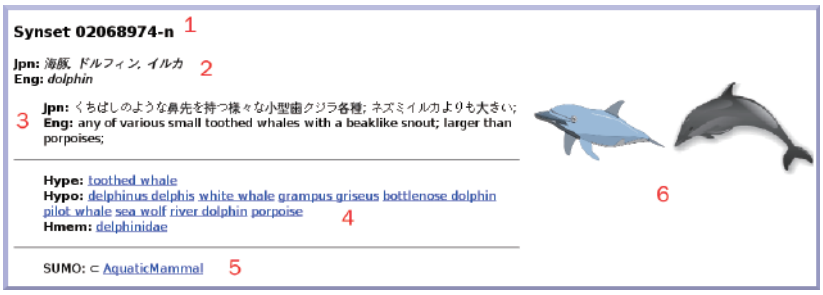
概念をどのように定義するのか

シンボル（文字列）に意味を与える方法がいくつかあります。

例えば、文法を利用する方法です。
　　クジラがプランクトンを食べた。
　　クジラ[名詞] が[助詞] プランクトン[名詞] を[助詞] 食べ[動詞] た[助動詞]。
この様に文法のパターンをいくつか用意すれば、それぞれの枠に入った文字列の意味を推定することができます。

概念をどのように定義するのか

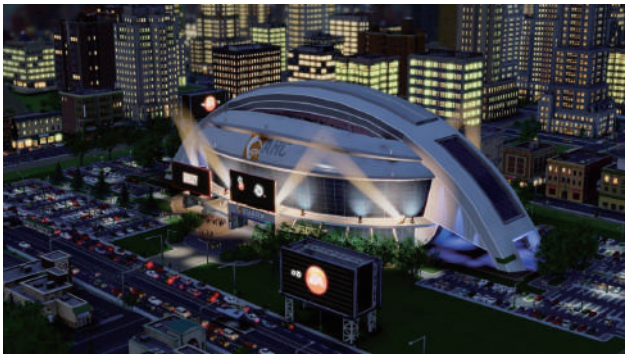
名詞動詞の関係性を整理した「概念辞書」というものを使用し、シンボル同士の関係性を定義することもできます。
下の例だと、イルカの上位概念（Hype）は[toothed whale（歯鯨）]で、下位概念（Hypo）は[delphinus delphis（マイルカ）]などが定義されています。



出展（情報通信研究機構）：http://compling.hss.ntu.edu.sg/wnja/

現代の技術で何が解決できそうか

文法や概念辞書でシンボルの意味付けをする場合、文法から外れた言い方や概念辞書にない記号（スラング、新出の言葉など）を全て網羅することはできません。
一つの解決策として、現実世界を仮想空間に表現し、その中でコンピュータに状況を理解させるという考え方があるようです。
地理、建物、食べ物などを完璧に再現できれば、文法によらずに対象物や状況を表現できるようになるかも知れません。



出展（シムシティ）：https://www.ea.com/ja-jp/games/simcity/simcity

演習：シンボルグラウンディング問題

あなたは喫茶店で働くロボットだとします。
お客さんが2人が喫茶店に入ってきました。

客A「俺はコーヒー。砂糖もお願い。」
客B「じゃあ、アイスで。」

と客2人は注文しました。
この注文を理解するために、どのような情報、判断プロセスが必要になるでしょうか？

チューリングテストとは

[Wikipediaより]
チューリングテスト（英: Turing test）は、アラン・チューリングが提案した、ある機械が「人間的」かどうかを判定するためのテストである。これが「知的であるかかどうか」とか「人工知能であるかどうか」とかのテストであるかどうかは、「知的」あるいは「(人工)知能」の定義、あるいは、人間が知的であるか、人間の能力は知能であるか、といった定義に依存する。

チューリングテストは「機械が人に近い振る舞いができるかどうかを判別すること」であり、それが知性か否かを判断するもの、とはされていません。

知性とは

知性とは何かを考えさせられる実験として、「中国語の部屋」という実験があります。

[Wikipedia]より
ある小部屋の中に、アルファベットしか理解できない人を閉じこめておく（例えば英国人）。この小部屋には外部と紙きれのやりとりをするための小さい穴がひとつ空いており、この穴を通して英国人に1枚の紙きれが差し入れられる。そこには彼が見たこともない文字が並んでいる。これは漢字の並びなのだが、英国人の彼にしてみれば、それは「★△◎▽☆口」といった記号の羅列にしか見えない。 彼の仕事はこの記号の列に対して、新たな記号を書き加えてから、紙きれを外に返すことである。どういう記号の列に、どういう記号を付け加えればいいのか、それは部屋の中にある1冊のマニュアルの中に全て書かれている。例えば"「★△◎▽☆口」と書かれた紙片には「■@◎▽」と書き加えてから外に出せ"などと書かれている。
彼はこの作業をただひたすら繰り返す。外から記号の羅列された紙きれを受け取り（実は部屋の外ではこの紙きれを"質問"と呼んでいる）、それに新たな記号を付け加えて外に返す（こちらの方は"回答"と呼ばれている）。すると、部屋の外にいる人間は「この小部屋の中には中国語を理解している人がいる」と考える。しかしながら、小部屋の中には英国人がいるだけである。彼は全く漢字が読めず、作業の意味を全く理解しないまま、ただマニュアルどおりの作業を繰り返しているだけである。それでも部屋の外部から見ると、中国語による対話が成立している。

演習：知性とは

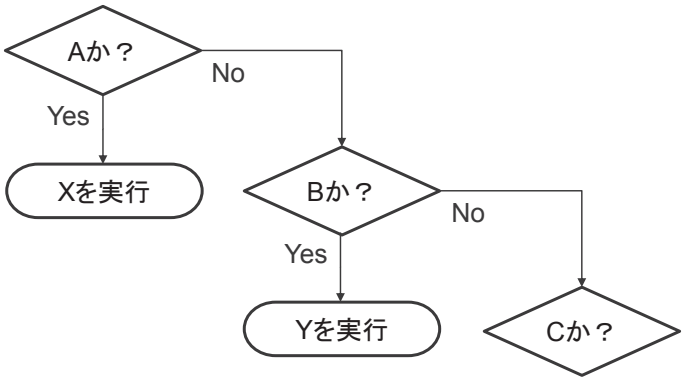
何ができたら知性とみなせるのか？
これは、そもそも「知性とはなにか？」という定義に関わる、非常に難しい問いです。

みなさんにとって、知性とは何でしょうか？

第3回：ルールベースの人工知能

ルールベースの人工知能

ルールベースとは、「どのような状況なら何を実行するのか」を明確に定義する方法です。
・もしAならXを実行する。AでないならYを実行する。
・もしAならXを実行する。BならYを実行する。AでもBでもなくCならZを実行する。



ルールをどのように構築するか

ルールベースシステムのアルゴリズムを構築するためには、システムが対象とする分野の知識が必要になります。

例えば「ホットプレートの温度管理」を考えます。
温度がある一定値（閾値）を超えたらスイッチを切って温度上昇を抑え、反対に閾値を下回ったらスイッチを入れ、温度を一定に保つ機能を実装する必要があるとします。

閾値はホットプレートの材質に依存するかも知れません。ホットプレートで何を作るか、に依存することもあるでしょう。
もしかしたら、安全に関わる法律も考慮する必要があるかも知れません。

このようにルールを構築するためには、対象となる分野の知識が必要になります。

演習：車のブレーキをルールベースで制御する

車の自動ブレーキ制御のプログラムを設計しているとします。
・車体の乾燥重量
・燃料の残量
・搭乗人数
・速度
・障害物までの距離
・ブレーキ踏み込み角度と制動力
など様々なパラメータを考慮する必要があると思われます。

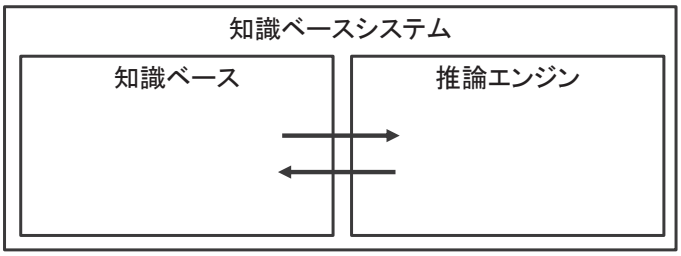
これらのパラメータ（上記以外で必要と思われるものも含め）を、どのような順番で判定することが好ましでしょうか。

知識ベースとは

[Wikipediaより]
知識ベース（knowledge base）はナレッジマネジメントのための特殊なデータベースであり、KBと略記されることもある。
それは知識の検索を可能とし、知識を組織化し、知識をコンピュータ上に集合させたものである。

知識ベースシステムとは

知識を利用して意思決定を支援するシステム、もしくは問題解決を支援するシステムを知識ベースシステムといいます。
知識ベースシステムは、知識を一定の形式で蓄積した知識ベースと、知識ベース内に蓄積された知識を活用して推論するための制御機構である推論エンジンで構成されます。



知識の表現

「～ならば、～である」というIF～THENで表現できる形で知識を蓄えていきます。
例えば、「肺炎か否か？」と判定するための知識として、以下のようなルールが考えられます。

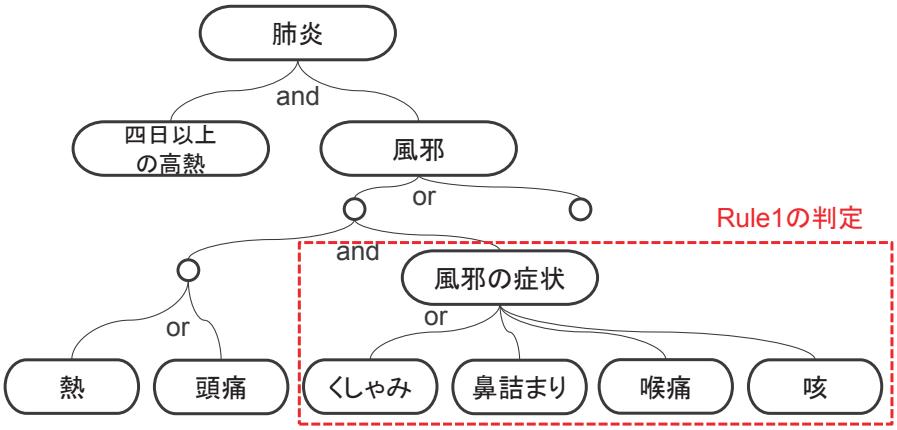
ルールNo	IF	THEN	確信度
Rule1	くしゃみがでる or 鼻詰まりがある or 喉が痛む or 咳が出る	風邪の症状がある	1.0
Rule2	（熱が高い or 頭がいたい）and 風邪の症状がある	風邪である	1.0
Rule3	（関節が痛む or 胃の調子が悪い）and 風邪の症状がある	風邪である	0.8
Rule4	四日以上高熱が続いている and 風邪である	肺炎の疑いがある	0.5

エキスパートシステムとは

エキスパートシステムは、知識ベースシステムの一種です。
問題領域のエキスパートの知識を用いて推論を行い、高度で複雑な問題をエキスパートと同等レベルで解決できるプログラムのことをエキスパートシステムといいます。

エキスパートシステムの例

例えば、肺炎か否かを判定するシステムがあるとします。
「知識の表現」のページに記載したルールの判定を行い、最終的に肺炎なのかを評価します。
ルールの判定の順番、確信度の取り扱いなどを推論エンジンが制御します。



エキスパートシステムと機械学習の共存

筆者は、「エキスパートシステムとして構築された金融機関のある不正取引検知システムを、機械学習を活用して高度化できないか？」という相談を受けたことがあります。

現行システムを調査し、また実データを用いてのPoC（概念実証）を実施した結果、「エキスパートシステムと機械学習のいいとこ取り」に落ち着くことになりました。

金融機関は規制当局の通達に従い規制をおこなうという、明確なルールベースに則る業務があります。また、現場のエキスパートの方が蓄積してきた知識ベースは、使用可能なデータのみですぐに機械学習で転用できるという類のものではありませんでした。

ただし、ある種のデータ群から不正利用を検知するという明確なタスクにおいて、機械学習の精度は圧倒的に人間を超えていました。

結局のところ、知識ベース（人間の知見と感の蓄積）でデータをグルーピングした後、それぞれのデータ群に対する不正検知については機械学習を適用する、というハイブリッド型の設計に落ち着きました。

演習：エキスパートシステムと機械学習

エキスパートシステムと機械学習が組み合わさった事例を調査してください。
どのような処理にエキスパートシステムを適用し、どのような処理に機械学習を適用しているのか整理比較してください。

第4回：画像処理

openCVとは

openCV（出展：https://ja.wikipedia.org/wiki/OpenCV）

- OpenCV（オープンシーヴィ、英語: Open Source Computer Vision Library）とはインテルが開発・公開したオープンソースのコンピュータビジョン向けライブラリ。2009年にWillow Garage（ウィロー・ガレージ）に開発が移管された後、2015年現在はItseezがメンテナンスを行なっている。なお、2016年5月26日にインテルがItseezを買収することが発表された。

コンピュータビジョン（出展：https://ja.wikipedia.org/wiki/コンピュータビジョン）

- コンピュータビジョン（computer vision）はコンピュータがデジタルな画像、または動画をいかによく理解できるか、ということを扱う研究分野である。工学的には、人間の視覚システムが行うことができるタスクを自動化することを追求する分野である。

openCVとは

OpenCVは下記のような関数を搭載しており、ビジョンの研究者や開発者の仕事を楽にするためのツールです。

- 線形代数や統計処理など，コンピュータビジョンに必要な各種数学関数
- 直線や曲線，テキストなど画像への描画関数
- OpenCVで使用したデータを読み込み/保存するための関数
- エッジ等の特徴抽出や画像の幾何変換，カラー処理等々の画像処理関数
- 物体追跡や動き推定などの動画像処理用関数
- 物体検出などのパターン認識関数
- 三次元復元のためのカメラ位置や姿勢の検出などのカメラキャリブレーション関数
- コンピュータにパターンを学習させるための機械学習関数
- 画像の読み込みや保存，表示，ビデオ入出力などインターフェース用関数

画像データの読み込み

- 画像を読み込むと3次元配列がピクセル分連なった行列に変換されます。
- 3次元配列の中身はBGRの順番となっています。

```
# 読み込みたい画像のパスを指定します。
path_img = 'resources/images/books.jpg'

"""
画像を読み込みます。読み込むと数値に変換されます。
3次元配列(BGR)がピクセル数連なった行列に変換されます。
"""
img = cv2.imread(path_img)

# 行列の中身を確認します。
print(img)
```

出展：https://github.com/piratefsh/image-processing-101/tree/master/images

```
[[[160 157 153]
  [158 155 150]
  [153 151 143]
  ...
  [133 161 162]
  [129 160 163]
  [129 163 163]]

 [[158 157 153]
  [155 155 149]
  [150 151 142]
  ...
```

色の要素の順番変更

- OpenCVでは画像はBGRフォーマットで読み込まれる初期設定となっていますが、Matplotlibの場合はRGBで読み込まれます。R：赤とB：青の数値が入れ替わっているまま表示させると、下図のように色調が異なっていることが確認できます。

出展：https://github.com/piratefsh/image-processing-101/tree/master/images

matplotlibによって「BGR」で表示

matplotlibによって「RGB」で表示

グレースケール化

- グレースケール画像とは、最も暗い色（黒）を表す数値を0、また最も明るい色（白）を表す数値を255とし、ピクセルの明るさを表現する256階調のスケールで、1つの色チャンネルしか持っていない画像のことです。

出展：https://github.com/piratefsh/image-processing-101/tree/master/images

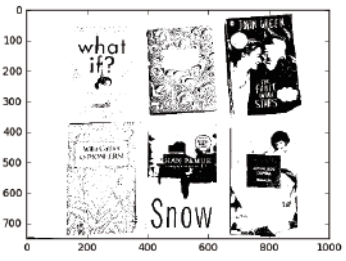
グレースケール化した画像

閾値処理

- ピクセルのチャンネル値がある閾値を超えた場合は、画像の各ピクセルを白いピクセルに、超えなかった場合は、黒いピクセルに置き換えます。
- 画像を2値画像、つまりシングルチャンネル画像に変換する処理で、画像セグメンテーションの一番簡単な方法です。
- グレースケール化はシングルチャンネル画像の一種です。

出展：https://github.com/piratefsh/image-processing-101/tree/master/images

グレースケール化した画像



閾値処理を実施した画像

画像のフィルタリング処理：畳み込み

- 学習データやテストデータにノイズ除去や画像のぼかしを入れると、汎化性能の高いロバスト（頑健な）なモデル作成や判定を行うことができます。
- openCVでは入力画像とカーネル(フィルタ)のconvolution（畳み込み）を計算できます。以下に5x5サイズの平均値フィルタに使うカーネルを示します。

$$K = \frac{1}{25}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

出展：http://labs.eecs.tottori-u.ac.jp/sd/Member/oyamada/OpenCV/html/py_tutorials/py_imgproc/py_filtering/py_filtering.html#filtering

画像のフィルタリング処理：ガウシアンフィルタ

ガウシアンフィルタ（出展：http://labs.eecs.tottori-u.ac.jp/sd/Member/oyamada/OpenCV/html/py_tutorials/py_imgproc/py_filtering/py_filtering.html#filtering）

- 注目画素との距離に応じて重みを変えるガウシアンカーネルをフィルタに使用する事もできます。
- カーネルの縦幅と横幅(どちらも奇数)に加え、ガウシアンの標準偏差値sigmaX(横方向)とsigmaY(縦方向)を指定する必要があります。sigmaXしか指定されなければ、sigmaYはsigmaXと同じだとみなされます。どちらの値も0にした場合、カーネルのサイズから自動的に計算されます。
- ガウシアンフィルタは白色雑音の除去に適しています。

OCRとは

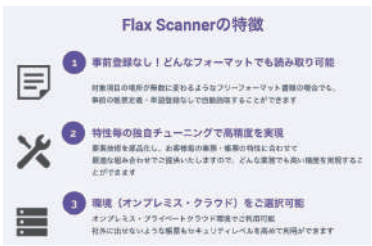
[Wikipediaより]
光学文字認識（こうがくもじにんしき、Optical character recognition）は、活字の文書の画像（通常イメージスキャナーで取り込まれる）を文字コードの列に変換するソフトウェアである。一般にOCRと略記される。OCRは、人工知能やマシンビジョンの研究分野として始まった。研究は続けられているが、OCRの中心はその実装と応用に移っている。紙に印刷された文書をデジタイズし、よりコンパクトな形で記録するのに必要とされる。さらに、文字コードに変換することで機械翻訳や音声合成の入力にも使えるようになり、テキストマイニングも可能となる。研究分野としては、パターン認識、人工知能、コンピュータビジョンが対応する。

AI-OCRの事例

近年、OCRに機械学習を応用したプロダクト、サービスを展開する企業が搭乗してきました。
定形フォーマットにおける読み取り精度の向上に注力する企業、非定形フォーマットでの読み取りに注力する企業など、各社独自の強みを発揮してAI-OCR市場を牽引しているようです。

出展（AI inside 株式会社）：https://inside.ai/

出展（株式会社 Cogent Labs）：https://www.tegaki.ai/



出展（株式会社シナモン）：https://cinnamon.is/flaxscanner

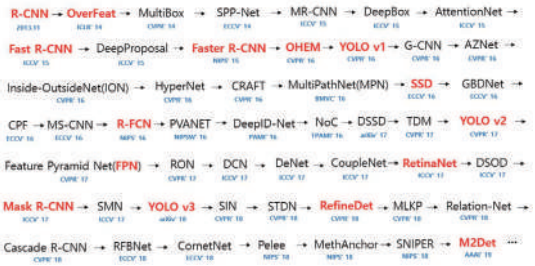
物体認識とは

物体認識は一般物体認識と特定物体認識に大別されます。

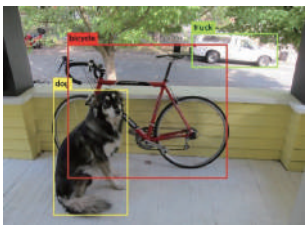
特定の対象物のみを認識するタスクは特定物体認識と呼ばれます。
一方、様々な対象物をまとめて認識するタスクは、一般物体認識と呼ばれます。

一般物体認識

画像中の物体の位置を検出し、かつ物体の名前を予測することです。
世界中で様々なアルゴリズムが開発されています。

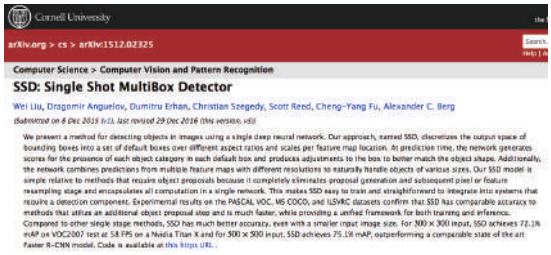


出展：https://github.com/hoya012/deep_learning_object_detection

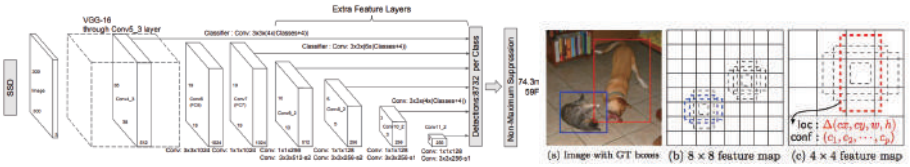

出展：https://pjreddie.com/darknet/yolo/

一般物体認識器の構築

一般物体認識について学習したい方は、例えば以下の論文をご参照ください。
https://arxiv.org/abs/1512.02325



判定領域のサイズを変えながらクラス分類をするタスクを繰り返し、モデルの重みを最適化します。



第5回：パターン検索

パターン検索とは

何かしらのデータを検索するときに、データ出現のパターンを利用して検索することをパターン検索といいます。
例えば、文字列のパターン検索には、以下のようなものがあります。

- 文字列一致
- 前方一致
- 後方一致

また、正規表現といわれる記述を使用し、以下のようなパターン検索もあります。

- 任意文字指定
- 部分文字指定
- 範囲指定

パターン検索：文字列一致

検索対象文字列と指定したキーワードが一致するかどうかを調べます。

（例）
検索対象文字列：
天気
検索文字列：
天気
検索結果：
True（検索文字列が、検索対象文字列と一致した。）

パターン検索：前方一致

検索対象文字列の先頭に、指定したキーワードが存在するかどうかを調べます。

（例）
検索対象文字列：
天気がよい
検索文字列：
天気
検索結果：
True（検索文字列が、検索対象文字列の先頭に存在した。）

パターン検索：後方一致

検索対象文字列の末尾に、指定したキーワードが存在するかどうかを調べます。

（例）
検索対象文字列：
天気が悪い一日
検索文字列：
一日
検索結果：
True（検索文字列が、検索対象文字列の末尾に存在した。）

正規表現とは

※本講義では、主にpythonを使用することを前提として学習していきます。

正規表現とは、文字列の集まりを1つの形式で表すための特別な書き方です。
例えば、以下のような正規表現があります。

メタ文字	説明	使用例	合致する文字列	合致しない文字列
.	任意の1文字	h.t	hat,hot,hit	head,height
*	直前のパターンを0回以上繰り返す	山川*	山,山川,山川川	里
+	直前のパターンを１回以上繰り返す	gre*n	greeeeeeen	grn
?	直前のパターンを0もしくは1回繰り返す	山川?	山,山川	山川川
{数字}	直前のパターンを数字の分だけ繰り返し	9{3}	999	9,99,9999
[文字,文字]	カッコ内の文字のいずれか１文字	[a,b,c]	a,b,c	d,e

特殊シーケンス	説明	例
\d	任意の数字	1,2,3など
\D	任意の数字以外	a,b,abcなど
\w	任意の英数字	1,2,333,ab,12abなど
\W	任意の英数字以外	山、川、空白など

参照：https://www.sejuku.net/blog/23232

パターン検索：任意文字指定

検索対象文字列に、任意の文字が入ったキーワードが含まれているかを調べます。

```
# 必要なライブラリをインポートします。
import re

# 検索する文字列です。猫山グループを検索します。
pattern = "猫山.*株式会社"

# 検索対象の文字列です。
text = "猫山株式会社"

# textの中からpatternに一致する文字を抽出します。
matchedList = re.findall(pattern, text)
if matchedList:
    print(matchedList)
else:
    print("検索結果なし")
```

['猫山株式会社']

```
# 検索対象の文字列です。
text = "猫山システムズ株式会社"

# textの中からpatternに一致する文字を抽出します。
matchedList = re.findall(pattern, text)
if matchedList:
    print(matchedList)
else:
    print("検索結果なし")
```

['猫山システムズ株式会社']

←

「猫山」と「株式会社」の間に「.*」という正規表現を記載しています。
「.」は任意の一文字という意味です。
「*」は直前のパターンを0回以上繰り返すという意味です。
「.*」と2つの正規表現を連続させることによって、任意の一文字が0回以上繰り返す、という意味になります。

よって「猫山」と「株式会社」の間に任意の文字が存在する（存在しない場合も含めて）パターンを抽出することができます。

パターン検索：部分文字指定

指定したキーワードの間に特定の文字列が入っているパターンを検索します。

```
# 検索する文字列です。
# 猫山グループのうち、システム会社とコンサルティング会社のみを検索します。
pattern = "猫山(システムズ|コンサルティング)株式会社"
```

「猫山」と「株式会社」の間に、「()」で囲んで「システムズ」と「コンサルティング」いう文字列を記載しています。
また、パイプと呼ばれる「|」はor（もしくは）の意味です。

「猫山」と「株式会社」の間に「システムズ」もしくは「コンサルティング」が存在する場合のみ検索することができます。

```
# 検索対象の文字列です。
text = "猫山システムズ株式会社"

# textの中からpatternに一致する文字を抽出します。
matchedList = re.findall(pattern, text)
if matchedList:
    print(matchedList)
else:
    print("検索結果なし")
```

```
['システムズ']
```

```
# 検索対象の文字列です。
text = "猫山コンサルティング株式会社"

# textの中からpatternに一致する文字を抽出します。
matchedList = re.findall(pattern, text)
if matchedList:
    print(matchedList)
else:
    print("検索結果なし")
```

```
['コンサルティング']
```

パターン検索：範囲指定

指定したキーワードの間に範囲指定した文字列が入っているパターンを検索します。

```
# 検索対象の文字列です。
text = "今日は朝の9時から11時まで散歩しました。歩いた距離は7kmでした。"
```

```
# 検索する文字列です。数値を取り出します。
#pattern = "\d"
# こちらの表現でも同様の動きをします。
pattern = "[0-9]"
```

「[0-9]」と記載すると、数字のみを検索することができます。
「＼d」でも同様の動きをします。

```
# コンパイルします。
re_compiled = re.compile(pattern)
```

```
# textの中からpatternに一致する文字を抽出します。
matchedList = re_compiled.findall(text)
if matchedList:
    print(matchedList)
```

```
['9', '1', '1', '7']
```

正規表現の組み合わせ

例えば、電話番号をパターンマッチングで取得することを考えます。
[https://sites.google.com/site/diveintopythonjp/home/7-seiki-hyougen/7-6-kesu-sutadi--denwa-bangou-no-koubun-kaiseki]

電話番号の記載のパターンを観察することで、正規表現を
複数組み合わせて検索することができます。
参考になるソースはwebですぐに見つけることができますので、
興味のある方は検索してみてください。

- 800-555-1212
- 800 555 1212
- 800.555.1212
- (800) 555-1212
- 1-800-555-1212
- 800-555-1212-1234
- 800-555-1212x1234
- 800-555-1212 ext. 1234
- work 1-(800) 555.1212 #1234

Example 7.10. 番号を見つけ出す

```
>>> phonePattern = re.compile(r'^(\d{3})-(\d{3})-(\d{4})$')  ❶
>>> phonePattern.search('800-555-1212').groups()  ❷
('800', '555', '1212')
>>> phonePattern.search('800-555-1212-1234')  ❸
>>>
```

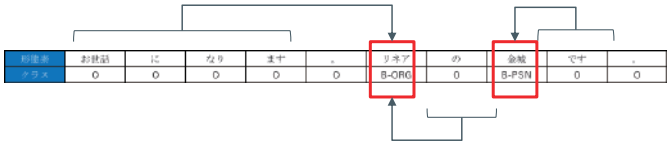
「意味」から文字列を取得する

言語の文法にしたがって文字列の出現パターンを解析するという方法もあります。
興味のある方は検索してみてください。
[https://qiita.com/Hironsan/items/326b66711eb4196aa9d4]

まず形態素解析を実施します。

```
(anaconda3-2.4.1) LINEAnoMacBook-puro:~ linea$ mecab
お世話になります。リネアの金城です。
お世話	名詞,サ変接続,*,*,*,*,お世話,オセワ,オセワ
に	助詞,格助詞,一般,*,*,*,に,ニ,ニ
なり	動詞,自立,*,*,五段・ラ行,連用形,なる,ナリ,ナリ
ます	助動詞,*,*,*,特殊・マス,基本形,ます,マス,マス
。	記号,句点,*,*,*,*,。,。,。
リネア	名詞,一般,*,*,*,*,*
の	助詞,連体化,*,*,*,*,の,ノ,ノ
金城	名詞,一般,*,*,*,*,金城,キンジョウ,キンジョー
です	助動詞,*,*,*,特殊・デス,基本形,です,デス,デス
。	記号,句点,*,*,*,*,。,。,。
EOS
```

CRF（条件付き確率場）で会社名と氏名を抽出します。



クラス	例
ART 固有物名	ノーベル文学賞、Windows7
LOC 地名	アメリカ、千葉県
ORG 組織	自民党、NHK
PSN 人名	安倍晋三、メルケル
DAT 日付	1月29日、2016/01/29
TIM 時間	午後三時、10:30
MNY 金額	241円、8ドル
PNT 割合	10%、3割

演習1：任意文字指定によるパターン認識

- 指定したキーワードの間に任意の文字列が入っているパターンの検索を実施してください。
- パターンに合致しない場合に検索できないことも確認してください。

演習2：部分文字指定によるパターン認識

- 指定したキーワードの間に特定の文字列が入っているパターンの検索を実施してください。
- パターンに合致しない場合に検索できないことも確認してください。

演習3：範囲指定によるパターン認識

- 指定したキーワードの間に数値が入っているパターンの検索を実施してください。
- パターンに合致しない場合に検索できないことも確認してください。

第7回：ゲーム理論

アジェンダ

- ゲーム理論
- ゲーム理論の例

ゲーム理論

ゲーム理論とは

[Wikipediaより]
ゲーム理論（ゲームりろん、英: game theory）とは、社会や自然界における複数主体が関わる意思決定の問題や行動の相互依存的状況を数学的なモデルを用いて研究する学問である。数学者ジョン・フォン・ノイマンと経済学者オスカー・モルゲンシュテルンの共著書『ゲームの理論と経済行動』(1944年) によって誕生した。元来は主流派経済学（新古典派経済学）への批判を目的として生まれた理論であったが、1980年代の「ゲーム理論による経済学の静かな革命」を経て、現代では経済学の中心的役割を担うようになった。

ゲーム理論の対象はあらゆる戦略的状況（英: strategic situations）である。「戦略的状況」とは自分の利得が自分の行動の他、他者の行動にも依存する状況を意味し、経済学で扱う状況の中でも完全競争市場や独占市場を除くほとんどすべてはこれに該当する。さらにこの戦略的状況は経済学だけでなく経営学、政治学、法学、社会学、人類学、心理学、生物学、工学、コンピュータ科学などのさまざまな学問分野にも見られるため、ゲーム理論はこれらにも応用されている。

囚人のジレンマ

2人の囚人AとBが逮捕されました。お互い独房に入れられ、意思疎通ができないとします。
二人に与えられた選択肢は「自白する」「自白しない」の二つです。
その際、警察はAとBそれぞれに以下のような条件を提示しました。

・AとBが両方自白しないなら、両方懲役1年となる。
・片方だけが自白した場合、自白した方は懲役なし、しなかった方は懲役5年となる。
・両方自白した場合、両方とも懲役3年となる。

2人の囚人AとBはどのような行動を取るでしょうか？

		囚人A	
		自白する	自白しない
囚人B	自白する	A：懲役3年 B：懲役3年	A：懲役5年 B：釈放
	自白しない	A：釈放 B：懲役5年	A：懲役1年 B：懲役1年

パレート最適

パレート最適とは、全体として最も利益が大きく最適な状態のことです。
囚人のジレンマにおいては、両者とも自白せず懲役1年ずつになる場合です。

囚人個人にとっては、「これ以上に利益を得ようと思ったら、誰か他の人が不利益を被る必要がある状態」とも言えます。
つまり、囚人AもしくはB個人の視点から考えると、懲役1年より利益を得る状態は、釈放されることです。
これが成り立つには、自分だけ自白し、相手は自白しないことが必要で、自白しなかった相手は懲役5年となってしまいます。

		囚人A	
		自白する	自白しない
囚人B	自白する	A：懲役3年 B：懲役3年	A：懲役5年 B：釈放
	自白しない	A：釈放 B：懲役5年	A：懲役1年 B：懲役1年

ナッシュ均衡

ナッシュ均衡とは、個々の最適を考えて合理的な判断をした結果ある均衡点に落ち着き、両者とも自分だけがそこから動く利益のない状態のことです。

囚人のジレンマの問題では、各々にとって最高の結果は自分の懲役が0年になることであり、最悪の結果は自分の懲役が5年になることです。
全体の最適化（パレート最適）を考えるとお互いに懲役1年になることがベストですが、2人は独房に入れられているため、話し合ってそれを到達する、というのは困難でしょう。
自分だけが懲役5年になるリスクを負わない合理的な行動（囚人AとBともに自白する）を取る可能性があります。

		囚人A	
		自白する	自白しない
囚人B	自白する	A：懲役3年 B：懲役3年	A：懲役5年 B：釈放
	自白しない	A：釈放 B：懲役5年	A：懲役1年 B：懲役1年

支配戦略

支配戦略とは、他のプレイヤーがどのような決断をしたとしても、自分のとるべき行動が決まってる状態のことです。
相手の選択に関わらず自分の行動が最適化されるため、選択を変える必要が無くなります。

囚人のジレンマの場合、相手が自白しようがしまいが、自分は自白をしておけば最も大きなリスクである懲役5年を避けられます。囚人のジレンマだと、両者ともに支配戦略を選択できる状況が生まれます。

お互いに支配戦略ができる状態を「支配戦略均衡」といい、支配戦略均衡はお互いに自らの選択を変える必要がなくなるため、ナッシュ均衡と同じ意味を持ちます。

		囚人A	
		自白する	自白しない
囚人B	自白する	A：懲役3年 B：懲役3年	A：懲役5年 B：釈放
	自白しない	A：釈放 B：懲役5年	A：懲役1年 B：懲役1年

社会的ジレンマ

自分にとって最も合理的だが周囲に非協力的な判断をした場合に最高の結果が得られる（相手は自白しないのに、自分だけ自白すると、自分は釈放され相手は懲役5年となる）。
しかし全員がその判断をした場合、全員で周囲に協力的な判断をした場合より悪い結果になる（2人とも自白すると、お互いに懲役3年になる）。

このような状況を社会的ジレンマといいます。

		囚人A	
		自白する	自白しない
囚人B	自白する	A：懲役3年 B：懲役3年	A：懲役5年 B：釈放
	自白しない	A：釈放 B：懲役5年	A：懲役1年 B：懲役1年

ゲーム理論の分類

ゲーム理論には協力ゲームと非協力ゲームがあります。

協力ゲームでは、参加者が協力し合うことでより多くの利益を獲得します。
個人の意思決定を優先するよりも、参加者全体の利益となる意思決定をした方が、かえって大きな利益を得られるというケースです。
例えばビジネスにおけるパートナー企業との協力が挙げられます。各社では獲得的なかった高利益率の案件を、パートナー企業と協力することによって獲得するような意思決定を行います。

非協力ゲームとは、参加者が競争をし合う状況を指します。
コモディティ化した市場における安売り競争などが挙げられます。

分析単位	協力ゲーム		非協力ゲーム
表現形式	提携型	戦略型	展開型

ゲーム理論の分類

ゲーム理論には協力ゲームと非協力ゲームがあります。

協力ゲームでは、参加者が協力し合うことでより多くの利益を獲得します。
個人の意思決定を優先するよりも、参加者全体の利益となる意思決定をした方が、かえって大きな利益を得られるというケースです。
例えばビジネスにおけるパートナー企業との協力が挙げられます。各社では獲得的なかった高利益率の案件を、パートナー企業と協力することによって獲得するような意思決定を行います。

非協力ゲームとは、参加者が競争をし合う状況を指します。
コモディティ化した市場における安売り競争などが挙げられます。

ゲーム理論の分類

非協力ゲームには標準形（戦略形）ゲームと展開形ゲームと言う2つの理論があります。

戦略形ゲームは、プレイヤーが同時に行動を選ぶ「同時ゲーム」で、代表的なゲームはじゃんけんなどです。
お互いに相手の行動を前もって知ることができず、同時に動きます。

展開形ゲームは、プレイヤーが順番で行動を選ぶ「交互ゲーム」で、多段階の意思決定を含みます。
代表的なゲームはチェスや将棋などです。

ゲーム理論の例

ゲーム理論の例：待ち合わせ

恋人と待ち合わせをしているとします。
その日は運悪く、二人ともスマートフォンを忘れてしまいました。
ですが、待ち合わせの時間に渋谷駅で待ち合わせ、という情報は覚えていたとします。

駅には多数の沿線が乗り入れていますし、改札も複数あります。連絡手段があれば、詳細に待ち合わせ場所をリアルタイムで話すことができるのですが、そうも行きません。

自分も相手も動き回ってなかなか会えない、ということになりかねません。
この時下手に二人とも動くよりは、片方だけが動いてもう片方を探し回る方が早く会える確率は高いです。しかし二人とも動かないという状況は最悪なため、二人ともそれを恐れ動き回ってしまいます。
この場合、片方だけが動いている状態がパレート最適、二人とも動いている状態がナッシュ均衡です。

ゲーム理論の例：偏差値

定期テストや入試などの成績は相対評価で行われます。
みんなが勉強しなくて点数が総じて低ければ、自分の点数が低くても相対評価は高くなります。
反対に、みんなが一生懸命勉強すれば、自分も必死に頑張らなければ評価が下がります。

みんながどれくらい頑張っているのだろう？と心配になったり、自分は全く勉強していないふりをして、隠れて必死に勉強するなど、駆け引きが生じてしまいます。

ゲーム理論の例：ホテルの宿泊料金設定

ある地域で人気グループのコンサートが開催されます。
地方から駆けつけるお客さんも多く、コンサート会場周辺の宿泊施設は、予約がいっぱいになる見込みです。

宿泊施設の協会に所属しているオーナーたちは、「お互いに宿泊料金を下げないようにしよう」と約束したものの、協会に所属していないホテルが価格を下げれば、先ずはそのホテルにお客が流れます。協会に所属しているオーナーたちは、コンサートの日が近づくに連れて、自分のホテルの空室率が気になって仕方ありません。

AIを駆使してコンサートチケットやホテル宿泊料金などを調整し、施設全体で収益を最大化しよう、という考え方があり、レベニューマネジメントと言われています。
ライバルホテルの宿泊料金、予約率がターゲットの日（コンサート日）に近づくにつれて変動する中で、自分のホテルの宿泊料金をどのように設定するのか、非常に面白い分野です。

演習：ゲーム理論の事例

皆さんの身近にあるゲーム理論の事例を挙げてください。

第8回：グラフ理論その1

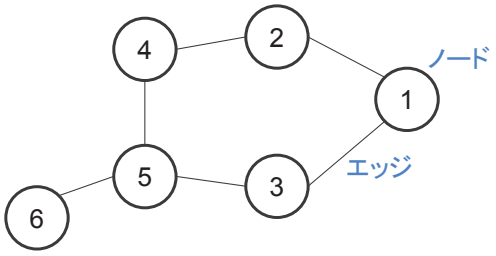
アジェンダ

- グラフ理論
- グラフ探索：ミニマックス法
- グラフ探索：ネガマックス法

グラフ理論

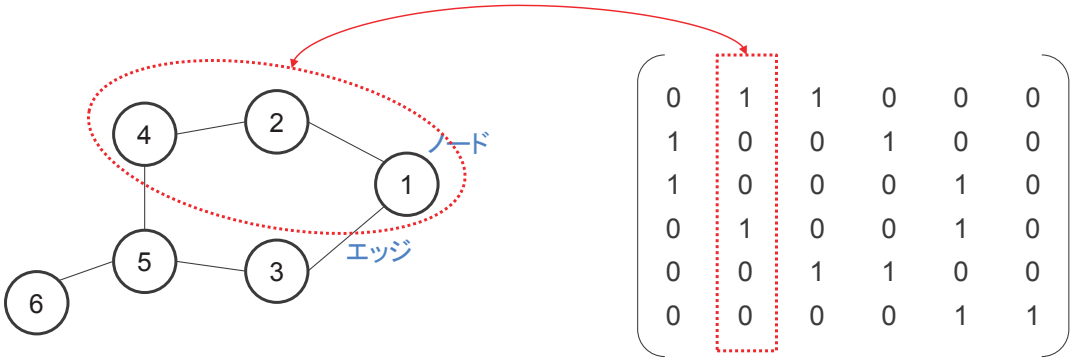
グラフ理論とは

グラフ理論(Graph theory)とは、ノード(節点・頂点)の集合とエッジ(枝・辺)の集合で構成されるグラフに関する数学の理論です。



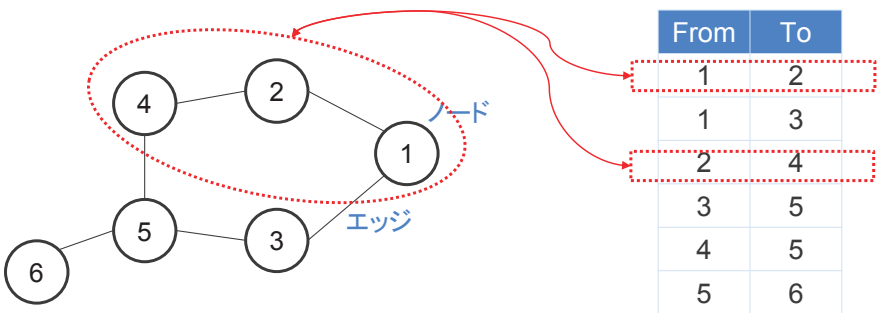
グラフ理論の表現方法

グラフの表現には、図ではなく行列を使う方法もあります。
下の例では、ノードの数6を辺とする6×6の行列で表現しています。



グラフ理論の表現方法

グラフを行列で表現した場合、ノードの数がNだとすると、必ずN^2の情報を保持することになります。
これでは、分析の際にコンピュータのリソースが枯渇してしまいます。
つながり（エッジ）が存在しないノードは無視し、つながりが存在するノードのみの情報で表現する方法もあります。

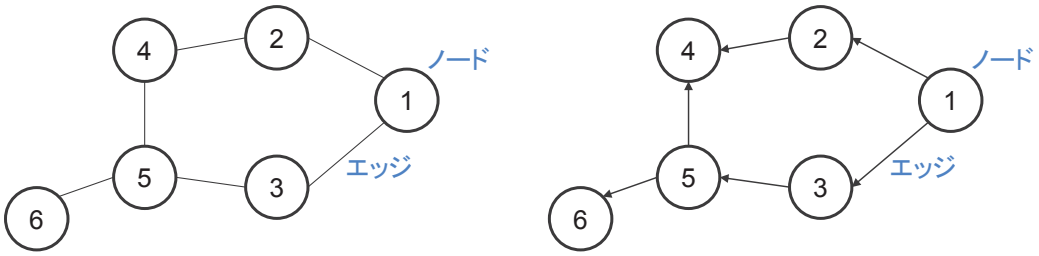


From	To
1	2
1	3
2	4
3	5
4	5
5	6

※From / Toと記載されていますが、エッジには方向がない無向グラフを例にしているため、[From:1, To:2]と[From:2, To:1]は同じ意味になります。

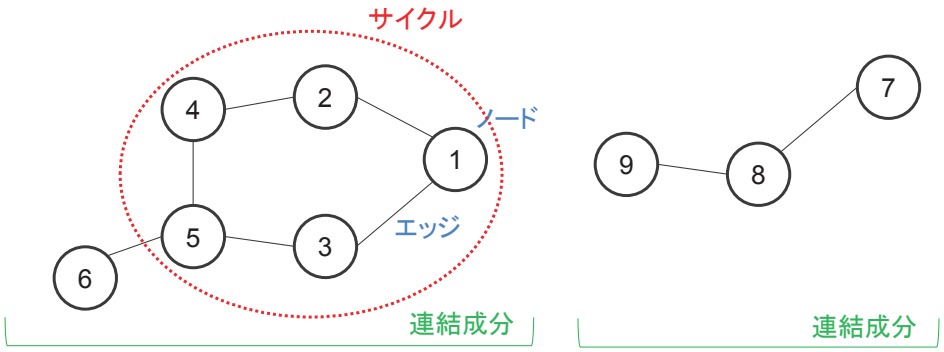
無向グラフと有向グラフ

エッジに向きがないグラフ(下左図)を無向グラフといいます。
エッジに向きがあるグラフ(下右図)を有向グラフといいます。



サイクル、連結成分

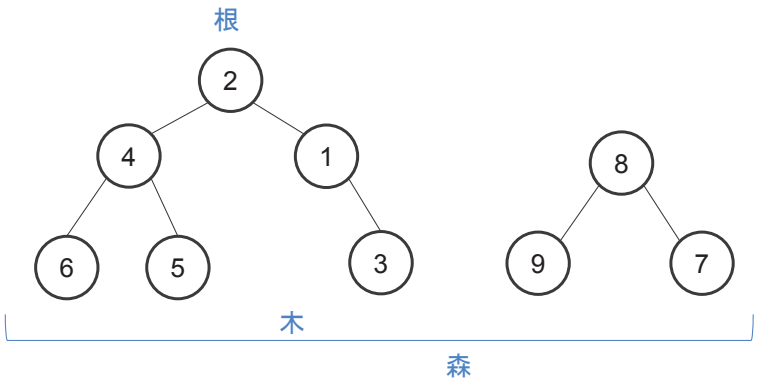
ノードとエッジを辿ると元に戻ってくる経路をサイクルといいます。
パスがつながっているかたまりを連結成分といいます。



木

連結でサイクルがないグラフを木といいます。
木が複数あるものを森といいます。

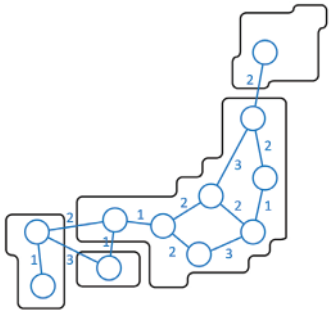
木は、あるノードを根として、そこからグラフがはじまると考えることがあります。



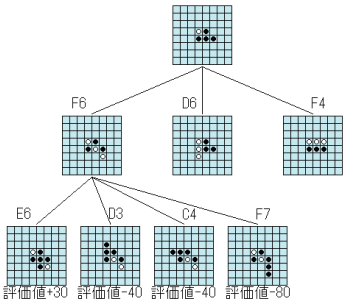
グラフ探索：MiniMax（ミニマックス）法

グラフ探索とは

ある状態や地点をノードで表現し、状態の遷移や地点間の移動をエッジで表現し、目的とする状態までのコストを最小とする経路を探すことを、グラフ探索といいます。



※参照（最短経路）：https://tajimarobotics.com/graph-theory-algorithm/

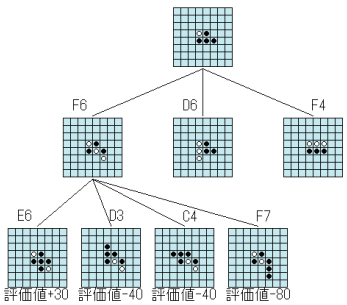


※参照（ゲーム木）：http://www.es-cube.net/es-cube/reversi/sample/html/2_3.html

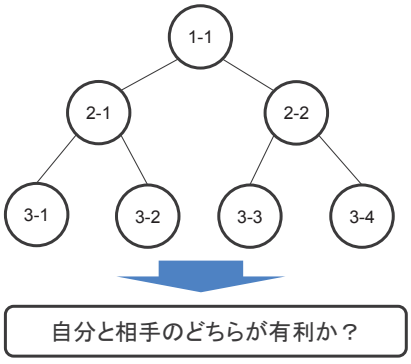
ミニマックス法

オセロを例としてミニマックス法を学習します。

オセロのように、自分にとっては最も有利な手を自分が打ち（max）、次に相手が自分にとって最も不利な手を打ち（min）、それらが交互に繰り返されるゲームの打ち手を考える際にゲーム木というものを利用します。

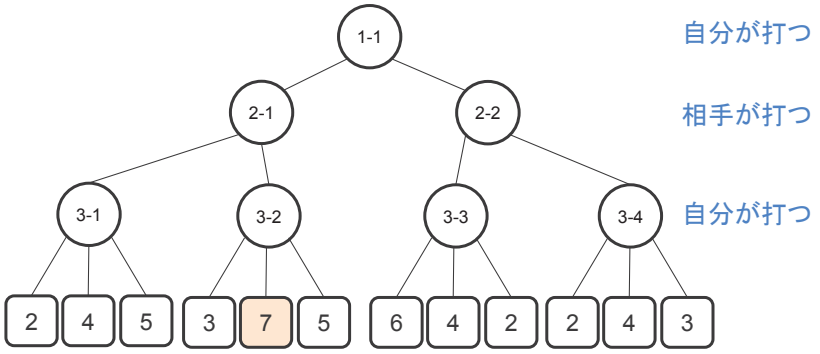


※参照（ゲーム木）：http://www.es-cube.net/es-cube/reversi/sample/html/2_3.html



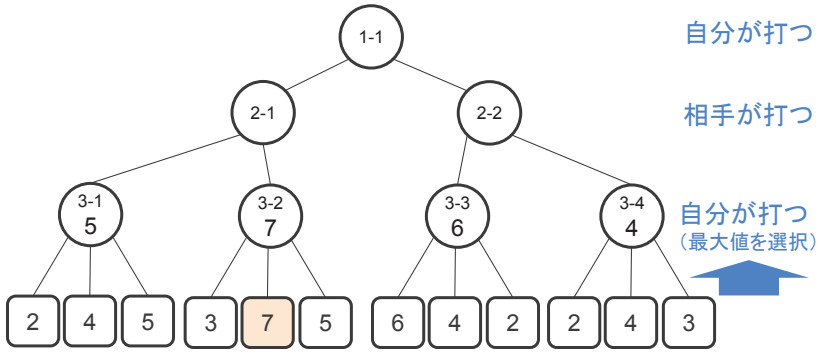
ミニマックス法

3手先までを考えてみます。
3手先までの全てのケースを想定してスコアリングした結果、自分が7の状態にあることが最良だとわかりました。スコア7にたどり着けるでしょうか？



ミニマックス法

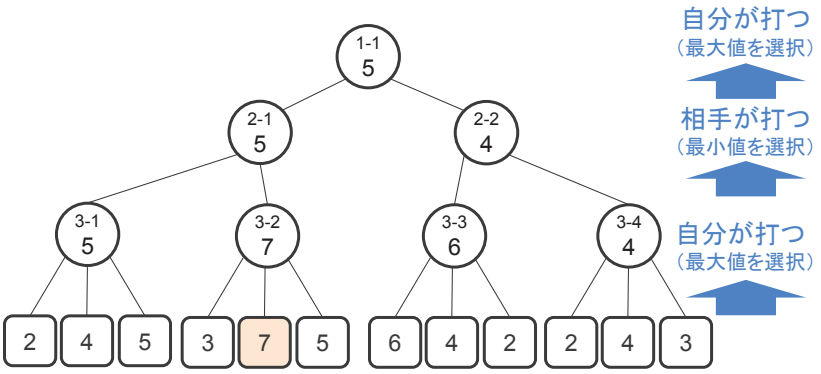
一番下から上に上がっていきます。
まず、3手目の自分の番では、スコアが最も大きくなるケースを選びます。



ミニマックス法

次に、2手目の相手の番では、スコアが最も低くなるケースが選ばれると想定します。
1手目の自分の番では、スコアが最も高くなるケースを選びます。

最終的に、自分が得られうる最良の状態は、スコア5で有ることがわかりました。

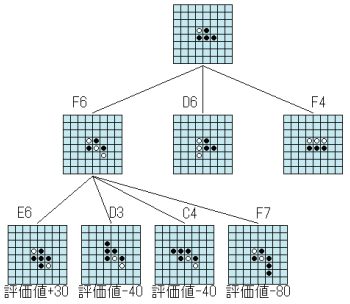


グラフ探索：NegaMax（ネガマックス）法

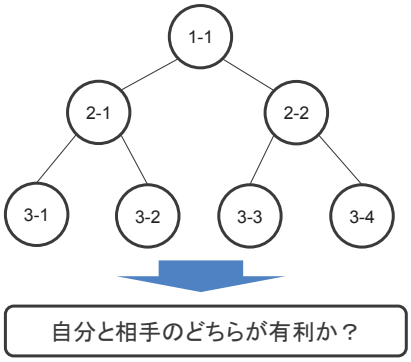
ネガマックス法

オセロを例としてネガマックス法を学習します。

ミニマックス法との違いは、相手はこちらの利益を最小にするように打つのではなく、相手自身の利益を最大にするように打つというところです。



※参照（ゲーム木）：http://www.es-cube.net/es-cube/reversi/sample/html/2_3.html

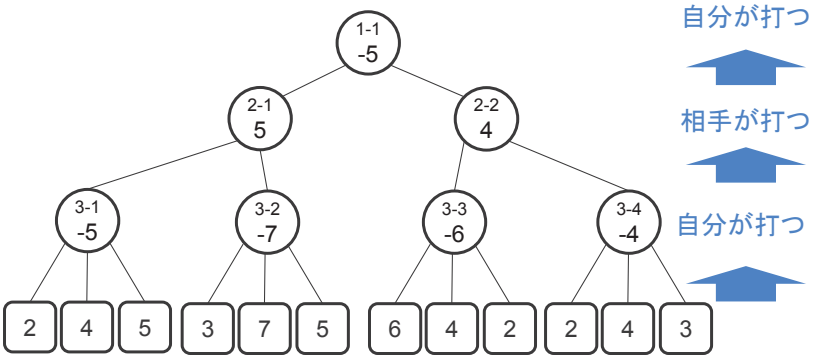


ネガマックス法

一番下の3手目から考えます。
自分に最も良い状態は、左から[5, 7, 6, 4]の状態になることです。
相手にとっては、-1を掛けた[-5, -7, -6, -4]となります。

相手は、相手のスコアが最大になるように行動すると仮定しますので、[-5, -4]を選択します。
自分にとっては-1を掛けた[5, 4]となります。

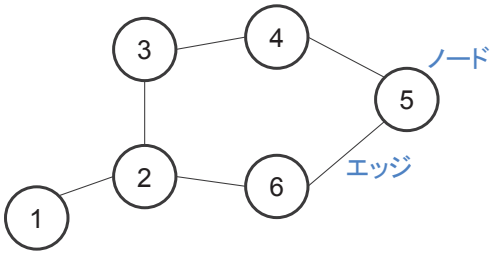
最終的に5を選択するので、自分にとっては
最良の状態はスコア5となります。



演習

演習1：グラフ理論の表現方法

左下に図で表現しているグラフを、From-Toの形式で表現してください。
ただし、無向グラフとします。

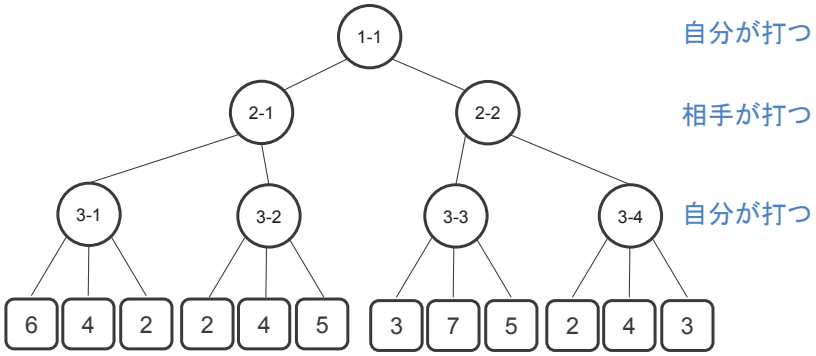


From	To
?	?
?	?
?	?
?	?
?	?
?	?

※From / Toと記載されていますが、エッジには方向がない無向グラフを例にしているため、[From:1, To:2]と[From:2, To:1]は同じ意味になります。

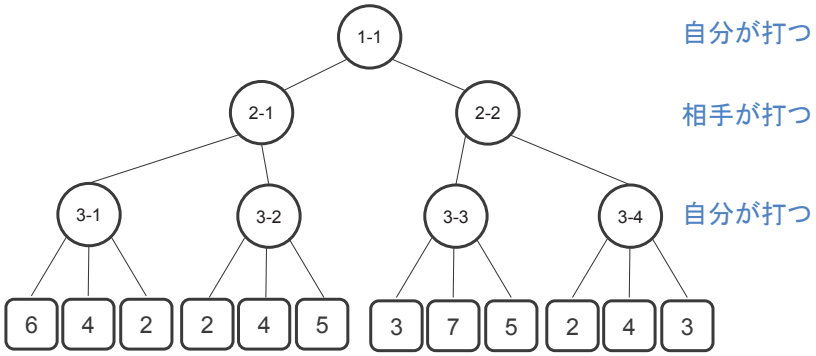
演習2：ミニマックス法

オセロの3手先を考えます。
ミニマックス法を使って、自分が得られるであろう最良のスコアを求めてください。



演習2：ネガマックス法

オセロの3手先を考えます。
ネガマックス法を使って、自分が得られるであろう最良のスコアを求めてください。



第9回：グラフ理論その2

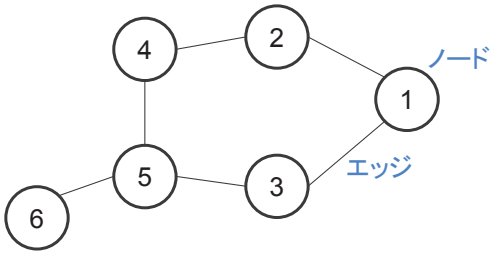
アジェンダ

- 前回の復習
- グラフ探索：α β（アルファ-ベータ）法

前回の復習：グラフ理論

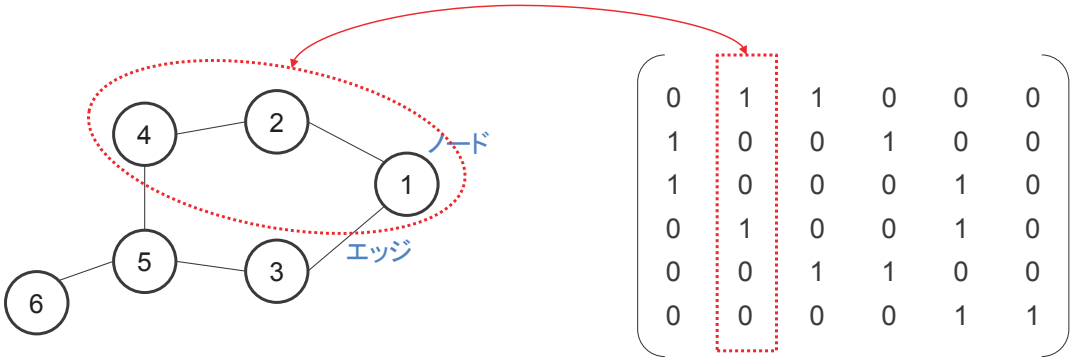
グラフ理論とは

グラフ理論（ Graph theory）とは、ノード（節点・頂点）の集合とエッジ（枝・辺）の集合で構成されるグラフに関する数学の理論です。



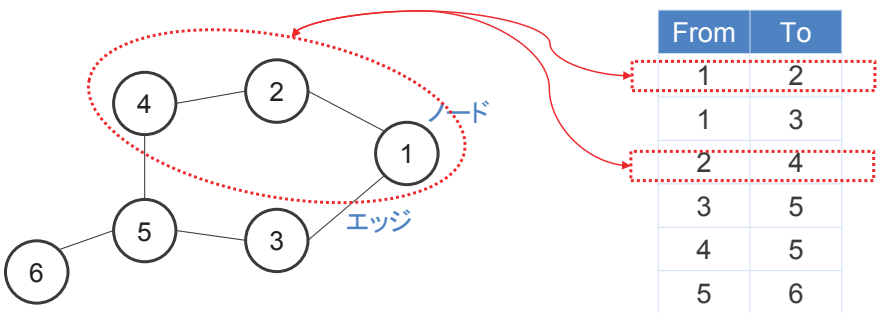
グラフ理論の表現方法

グラフの表現には、図ではなく行列を使う方法もあります。
下の例では、ノードの数6を辺とする6×6の行列で表現しています。



グラフ理論の表現方法

グラフを行列で表現した場合、ノードの数がNだとすると、必ずN^2の情報を保持することになります。
これでは、分析の際にコンピュータのリソースが枯渇してしまいます。
つながり（エッジ）が存在しないノードは無視し、つながりが存在するノードのみの情報で表現する方法もあります。

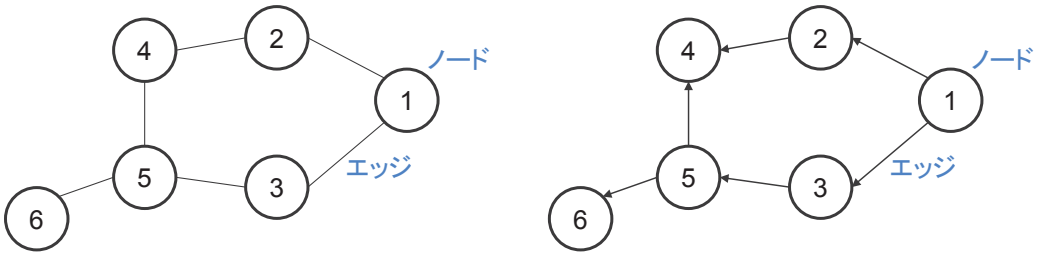


From	To
1	2
1	3
2	4
3	5
4	5
5	6

※From / Toと記載されていますが、エッジには方向がない無向グラフを例にしているため、[From:1, To:2]と[From:2, To:1]は同じ意味になります。

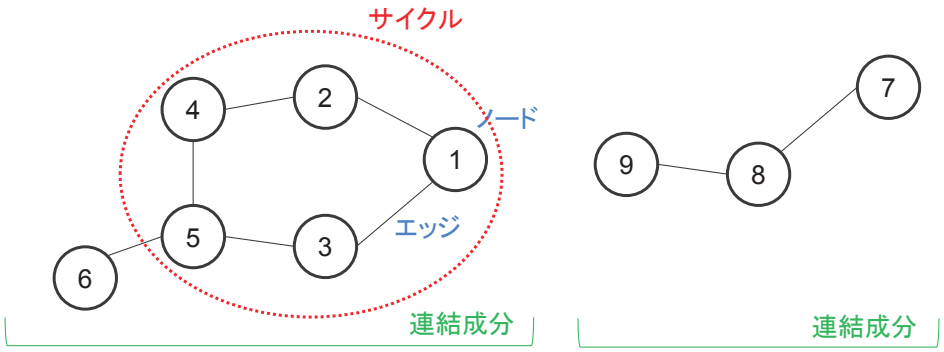
無向グラフと有向グラフ

エッジに向きがないグラフ(下左図)を無向グラフといいます。
エッジに向きがあるグラフ(下右図)を有向グラフといいます。



サイクル、連結成分

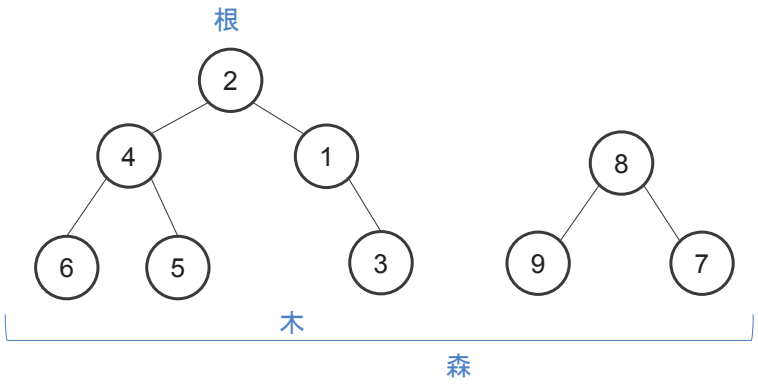
ノードとエッジを辿ると元に戻ってくる経路をサイクルといいます。
パスがつながっているかたまりを連結成分といいます。



木

連結でサイクルがないグラフを木といいます。
木が複数あるものを森といいます。

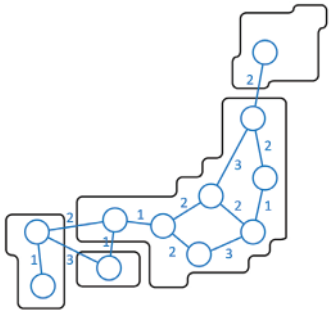
木は、あるノードを根として、そこからグラフがはじまると考えることがあります。



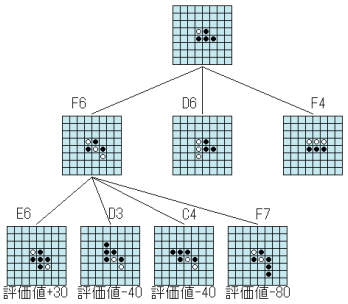
前回の復習：グラフ探索：MiniMax（ミニマックス）法

グラフ探索とは

ある状態や地点をノードで表現し、状態の遷移や地点間の移動をエッジで表現し、目的とする状態までのコストを最小とする経路を探すことを、グラフ探索といいます。



※参照（最短経路）：https://tajimarobotics.com/graph-theory-algorithm/

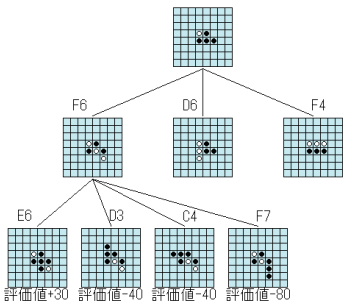


※参照（ゲーム木）：http://www.es-cube.net/es-cube/reversi/sample/html/2_3.html

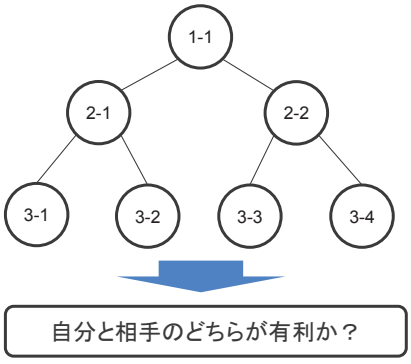
ミニマックス法

オセロを例としてミニマックス法を学習します。

オセロのように、自分にとっては最も有利な手を自分が打ち（max）、次に相手が自分にとって最も不利な手を打ち（min）、それらが交互に繰り返されるゲームの打ち手を考える際にゲーム木というものを利用します。

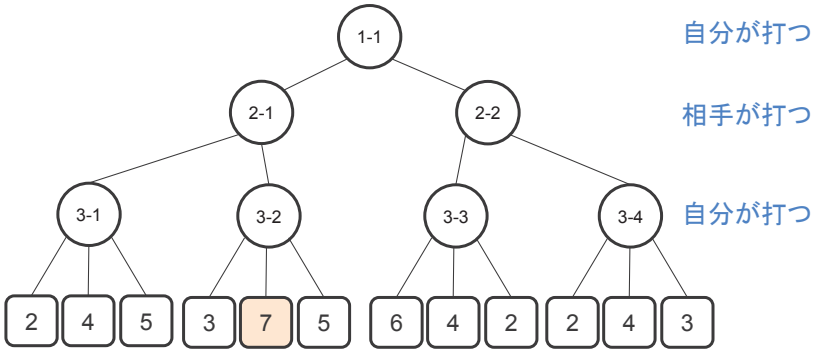


※参照（ゲーム木）：http://www.es-cube.net/es-cube/reversi/sample/html/2_3.html



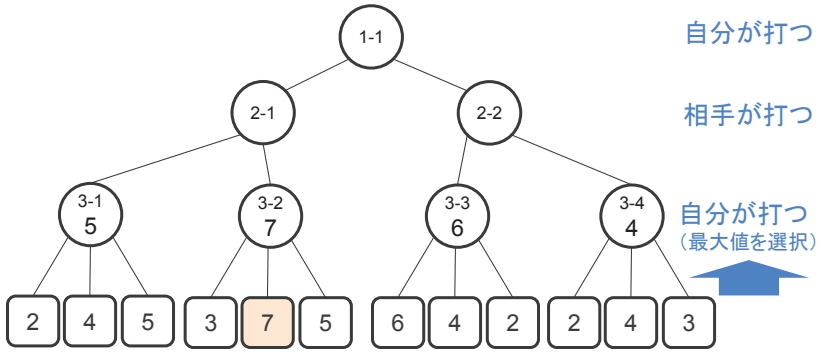
ミニマックス法

3手先までを考えてみます。
3手先までの全てのケースを想定してスコアリングした結果、自分が7の状態にあることが最良だとわかりました。スコア7にたどり着けるでしょうか？



ミニマックス法

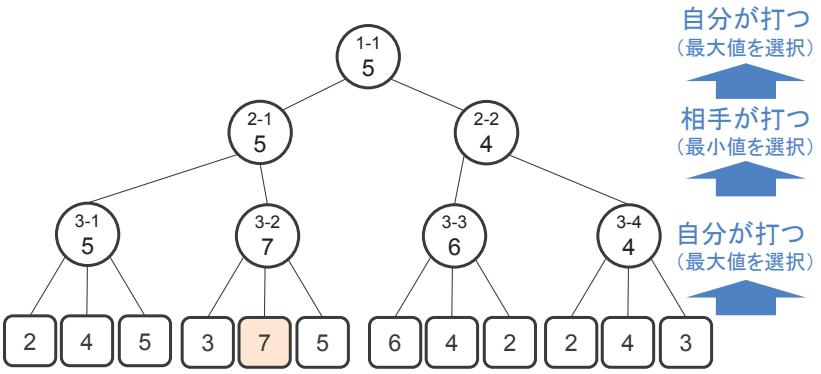
一番下から上に上がっていきます。
まず、3手目の自分の番では、スコアが最も大きくなるケースを選びます。



ミニマックス法

次に、2手目の相手の番では、スコアが最も低くなるケースが選ばれると想定します。
1手目の自分の番では、スコアが最も高くなるケースを選びます。

最終的に、自分が得られうる最良の状態は、スコア5で有ることがわかりました。

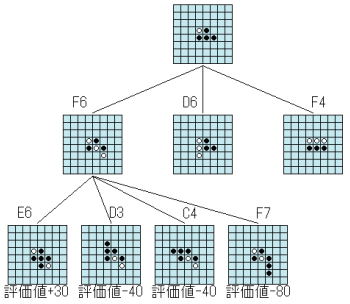


前回の復習：グラフ探索：NegaMax（ネガマックス）法

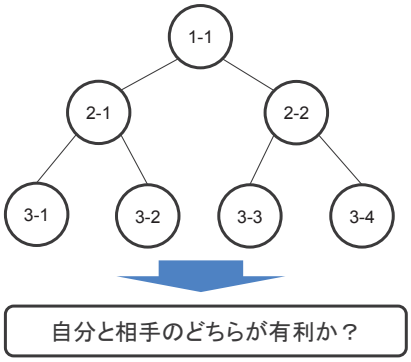
ネガマックス法

オセロを例としてネガマックス法を学習します。

ミニマックス法との違いは、相手はこちらの利益を最小にするように打つのではなく、相手自身の利益を最大にするように打つというところです。



※参照（ゲーム木）：http://www.es-cube.net/es-cube/reversi/sample/html/2_3.html

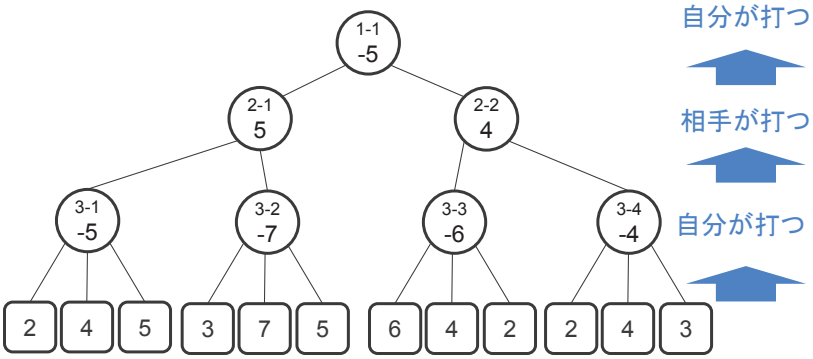


ネガマックス法

一番下の3手目から考えます。
自分に最も良い状態は、左から[5, 7, 6, 4]の状態になることです。
相手にとっては、-1を掛けた[-5, -7, -6, -4]となります。

相手は、相手のスコアが最大になるように行動すると仮定しますので、[-5, -4]を選択します。
自分にとっては-1を掛けた[5, 4]となります。

最終的に5を選択するので、自分にとっては
最良の状態はスコア5となります。

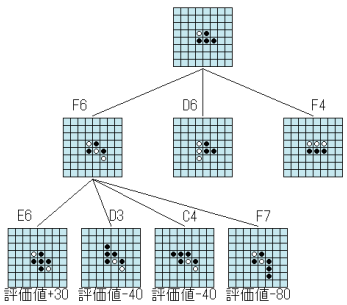


グラフ探索：αβ（アルファ-ベータ）法

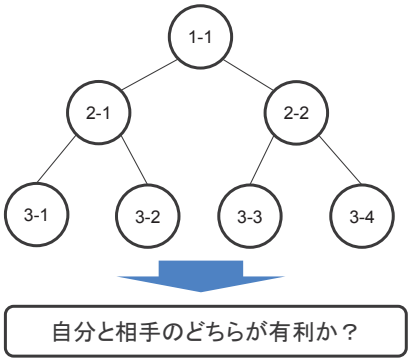
アルファ-ベータ法

アルファ-ベータ法は、ミニマックス法を改良したアルゴリズムです。

全ての手を探索すると膨大な時間を要するため、探索ノード数を削減するように工夫したアルゴリズムです。



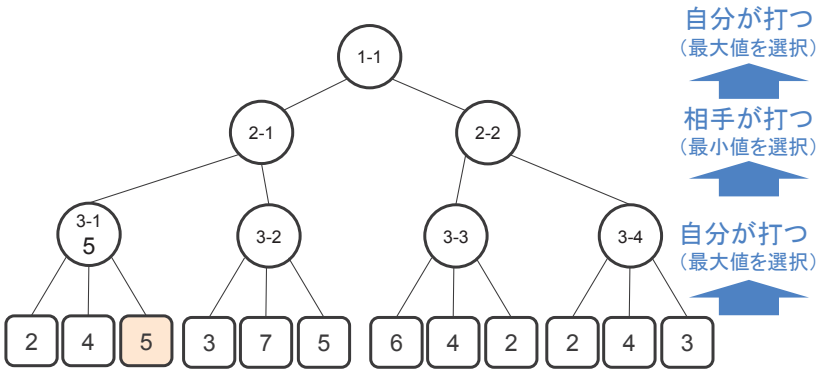
※参照（ゲーム木）：http://www.es-cube.net/es-cube/reversi/sample/html/2_3.html



アルファ-ベータ法

各ノードで、必ず左側から探索するとします。

まず[1-1] -> [2-1] -> [3-1]と探索します。
[3-1]に属している数字で最も大きな数字は5です。

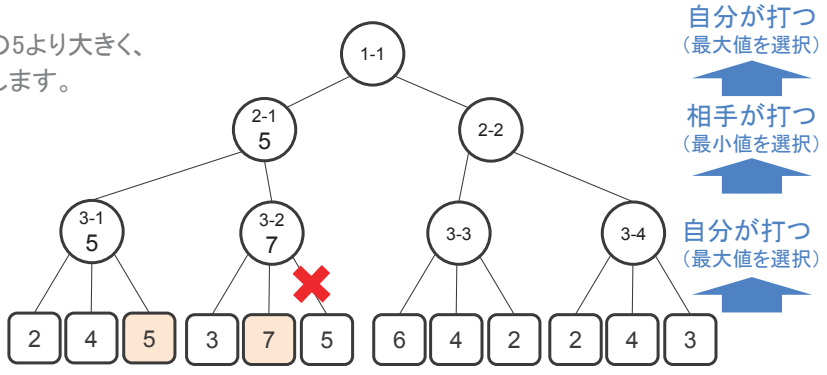


アルファ-ベータ法

次に[1-1] -> [2-1] -> [3-2]と探索します。
[3-2]に属している数字を左から見ていきます。

ミニマックス法では、[3-2]において3, 7, 5から最大の数字を選び、
[2-1]においては[3-1]と[3-2]で小さい数字を選びます。

[3-2]において7が見つかった時点で、[3-1]の5より大きく、
[2-1]では[3-1]の5が選択されることが決定します。
よって[3-2]において5の探索はしません。



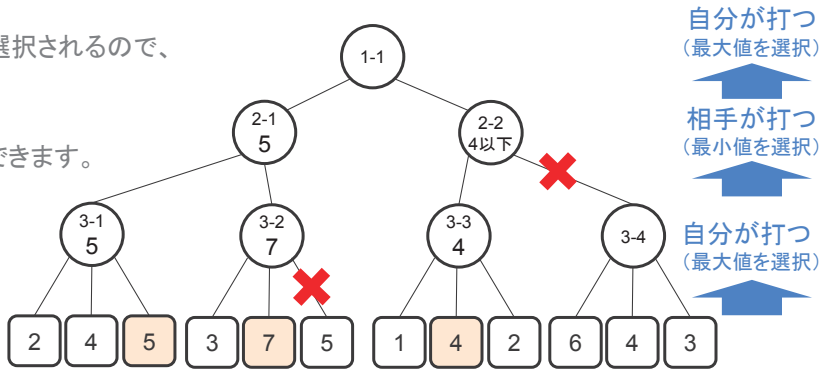
アルファ-ベータ法

次に[1-1] -> [2-2] -> [3-3]と探索します。
[3-3]において最大値の4が選択されます。

[2-2]は、[3-3]と[3-4]の小さい値が選択されます。
[3-3]において4となったということは、[2-2]は4以下であるということです。

[1-1]においては[2-1]と[2-2]の大きい値が選択されるので、
[2-1]の5が選択されます。

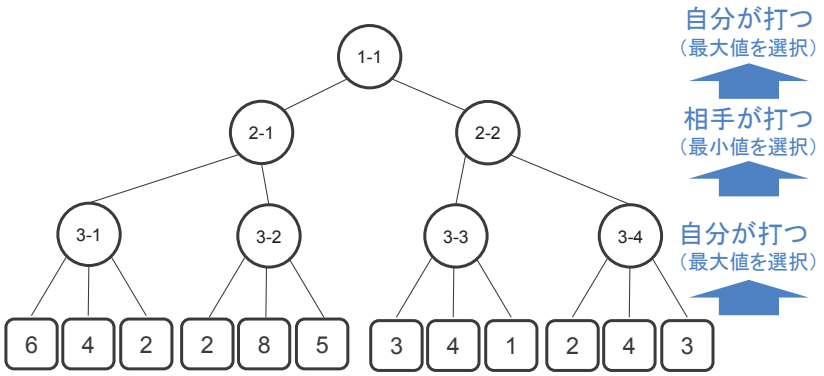
よって、[3-4]以下の探索は省略することができます。



演習

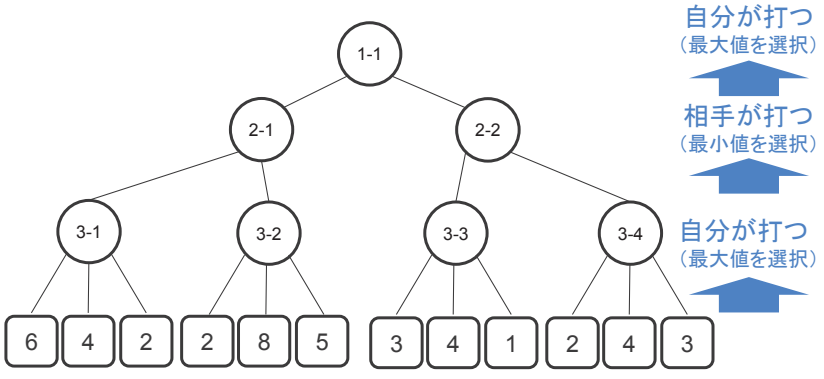
演習1：ミニマックス法

オセロの3手先を考えます。
ミニマックス法を使って、自分が得られるであろう最良のスコアを求めてください。



演習2：アルファ-ベータ法

オセロの3手先を考えます。
アルファ-ベータ法を使って、自分が得られるであろう最良のスコアを求めてください。



第10回：グラフ理論その3

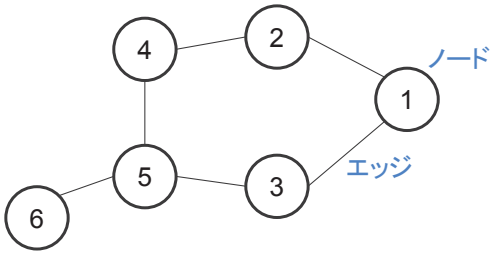
アジェンダ

- 前回までの復習
- 深さ優先探索
- 幅優先探索
- A*探索
- 動的計画法

前回の復習：グラフ理論

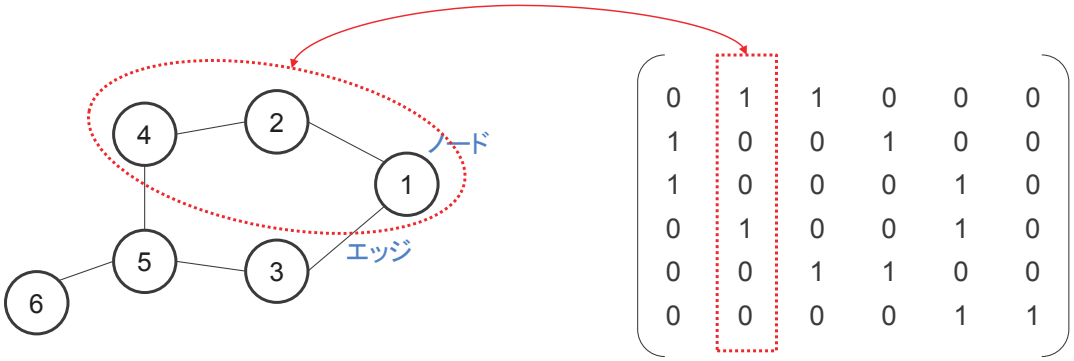
グラフ理論とは

グラフ理論（ Graph theory）とは、ノード（節点・頂点）の集合とエッジ（枝・辺）の集合で構成されるグラフに関する数学の理論です。



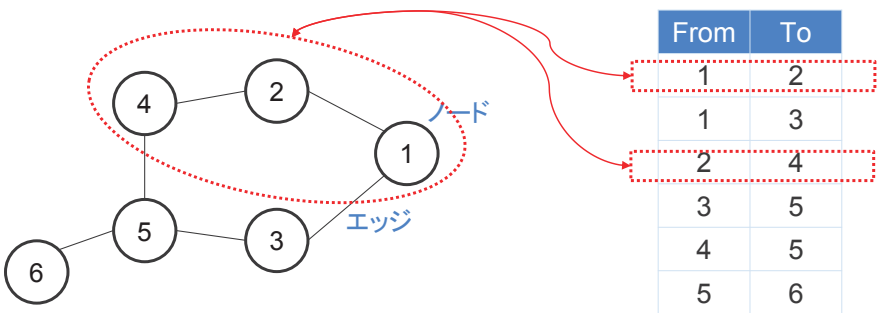
グラフ理論の表現方法

グラフの表現には、図ではなく行列を使う方法もあります。
下の例では、ノードの数6を辺とする6×6の行列で表現しています。



グラフ理論の表現方法

グラフを行列で表現した場合、ノードの数がNだとすると、必ずN^2の情報を保持することになります。
これでは、分析の際にコンピュータのリソースが枯渇してしまいます。
つながり（エッジ）が存在しないノードは無視し、つながりが存在するノードのみの情報で表現する方法もあります。

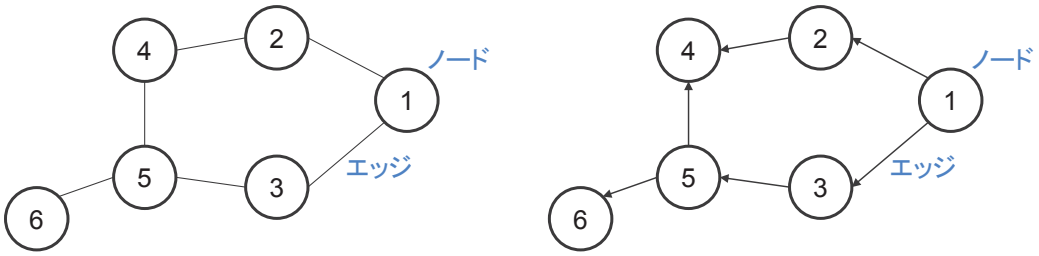


From	To
1	2
1	3
2	4
3	5
4	5
5	6

※From / Toと記載されていますが、エッジには方向がない無向グラフを例にしているため、[From:1, To:2]と[From:2, To:1]は同じ意味になります。

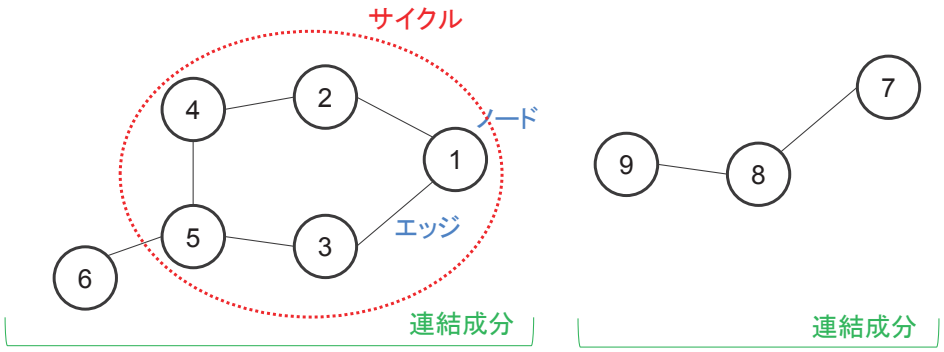
無向グラフと有向グラフ

エッジに向きがないグラフ(下左図)を無向グラフといいます。
エッジに向きがあるグラフ(下右図)を有向グラフといいます。



サイクル、連結成分

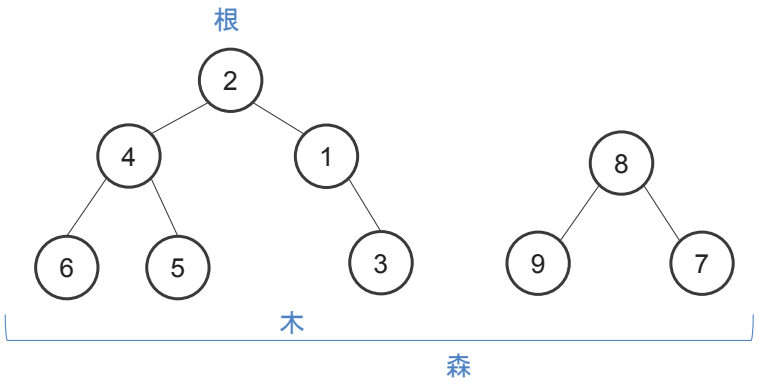
ノードとエッジを辿ると元に戻ってくる経路をサイクルといいます。
パスがつながっているかたまりを連結成分といいます。



木

連結でサイクルがないグラフを木といいます。
木が複数あるものを森といいます。

木は、あるノードを根として、そこからグラフがはじまると考えることがあります。



深さ優先探索

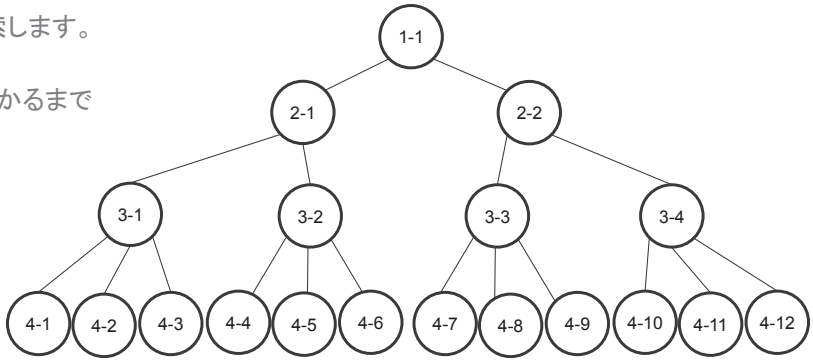
深さ優先探索

深さ優先探索とは、先ずは行けるところまでまっすぐ探索し、行き止まりになったら1つ手前に戻り別のエッジを探索するという手法です。

左側から探索するとすると、[1-1]->[2-1]->[3-1]->[4-1]と進みます。
[4-1]で行き止まりになるので[3-1]に戻り、[4-2][4-3]を探索します。

次に[2-1]まで戻り、[2-1]->[3-2]->[4-4]と探索します。

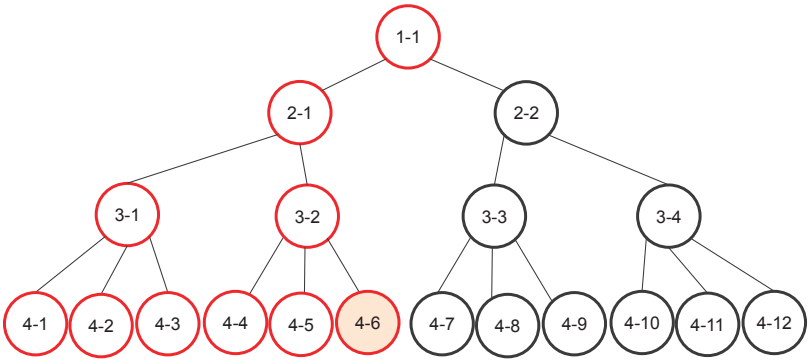
このような探索の仕方を、目的のノードが見つかるまで繰り返します。



深さ優先探索の計算量

深さ優先探索の計算量を考えてみます。
[4-6]を見つけるまで、何個のノードを探索する必要があるでしょうか。

左側から探索するとした場合、[4-6]は10個目のノードとして見つかります。



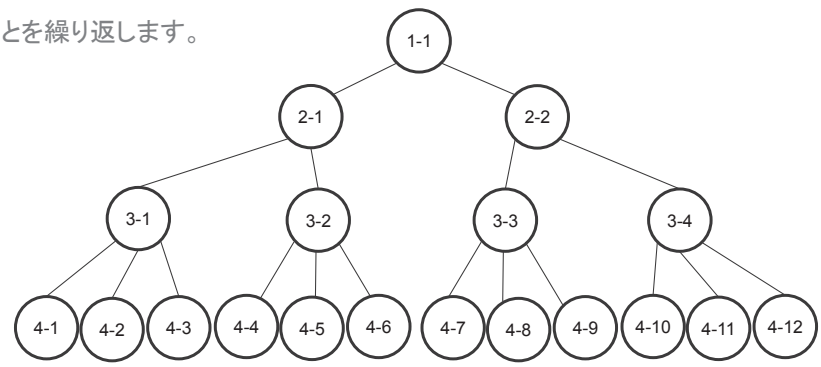
幅優先探索

幅優先探索

幅優先探索とは、出発点に近い点から探索するという手法です。

左側から探索するとすると、[1-1]に隣接している[2-1]->[2-2]と進みます。
次に、[2-1]に隣接している [3-1][3-2]を探索します。
次に、[2-2]に隣接している [3-3][3-4]を探索します。

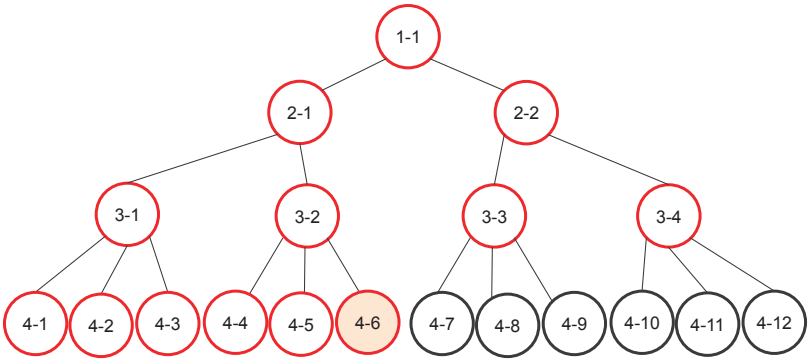
このような出発点から横方向に探索していくことを繰り返します。



幅優先探索の計算量

幅優先探索の計算量を考えてみます。
[4-6]を見つけるまで、何個のノードを探索する必要があるでしょうか。

左側から探索するとした場合、[4-6]は13個目のノードとして見つかります。

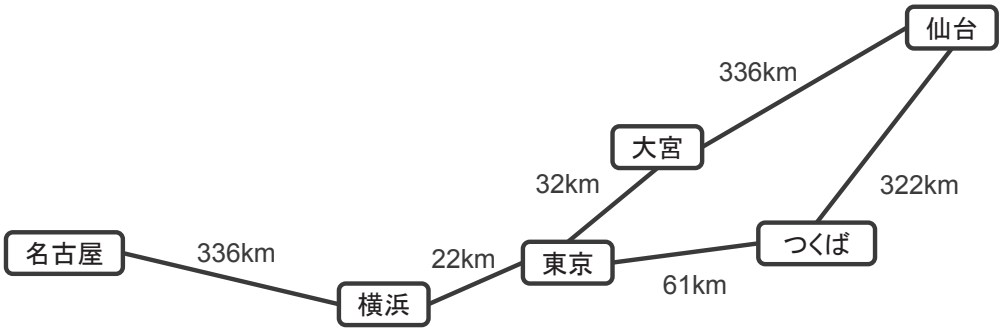


A*探索

A*（エースター）探索

[Wikipediaより]
A* アルゴリズムは、「グラフ上でスタートからゴールまでの道を見つける」というグラフ探索問題において、ヒューリスティック関数 h(n) という探索の道標となる関数を用いて探索を行うアルゴリズムである。

例えば、東京から仙台まで旅行に行くことを考えます。
どのルートを辿ればよいでしょうか？



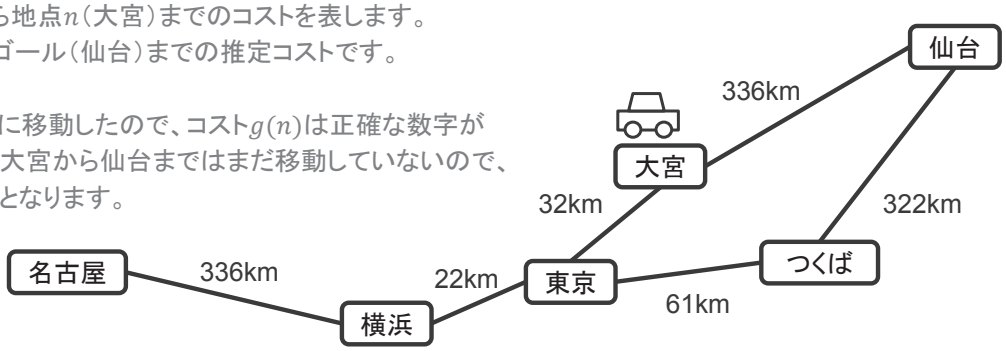
A*（エースター）探索

A*探索では、コスト関数$f(n)$を考えます。
もし最短距離で到着したい人は、コスト関数は最短距離を返す関数となります。
距離でなく時間を優先したい人は、最短時間を返す関数となります。

$$f(n) = g(n) + h(n)$$

例えば東京を出発し仙台を目指すとします。現在は大宮にいるとします。
$g(n)$はスタート（東京）から地点n（大宮）までのコストを表します。
$h(n)$は地点n（大宮）からゴール（仙台）までの推定コストです。

※東京から大宮まで実際に移動したので、コスト$g(n)$は正確な数字がわかっています。ですが、大宮から仙台まではまだ移動していないので、$h(n)$はあくまで推定コストとなります。



A*（エースター）探索

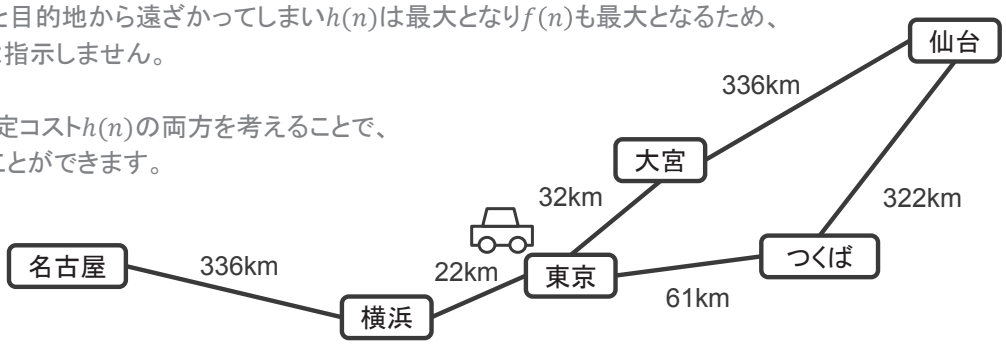
A*探索で、どのように経路を選択するのか見てみましょう。コスト関数は最短距離を返す関数とします。

$$f(n) = g(n) + h(n)$$

また、推定コスト$h(n)$は各都市の緯度経度から計算したユークリッド距離$\sqrt{x^2 + y^2}$を考えるとします。

スタートの東京にいるあなたが、横浜/大宮/つくばに移動した3つのパターンを考えてみます。
$g(n)$のみを考慮すれば、東京から横浜に移動したパターンが、コストは最小（22km）になります。
ですが、横浜に移動すると目的地から遠ざかってしまい$h(n)$は最大となり$f(n)$も最大となるため、
ナビは横浜に行くようには指示しません。

このようにコスト$g(n)$と推定コスト$h(n)$の両方を考えることで、
最適なルートを検索することができます。



動的計画法

動的計画法とは

[Wikipediaより]
動的計画法（どうてきけいかくほう、英: Dynamic Programming, DP）は、計算機科学の分野において、アルゴリズムの分類の1つである。対象となる問題を複数の部分問題に分割し、部分問題の計算結果を記録しながら解いていく手法を総称してこう呼ぶ。
細かくアルゴリズムが定義されているわけではなく、下記2条件を満たすアルゴリズムの総称である。

- 帰納的な関係の利用：より小さな問題例の解や計算結果を帰納的な関係を利用してより大きな問題例を解くのに使用する。
- 計算結果の記録：小さな問題例、計算結果から記録し、同じ計算を何度も行うことを避ける。帰納的な関係での参照を効率よく行うために、計算結果は整数、文字やその組みなどを見出しにして管理される。

以下の2種類の実現方法がある。

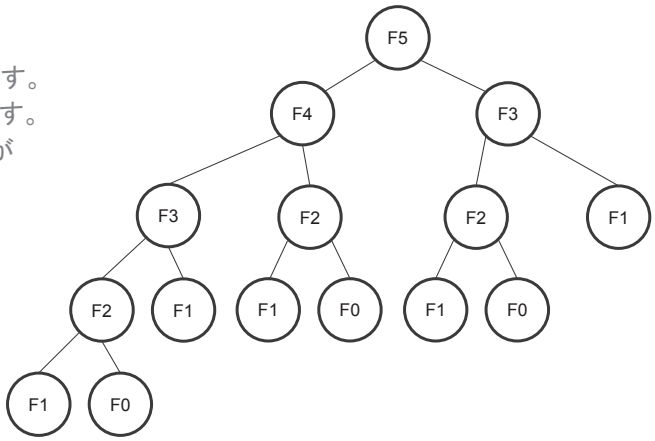
- 履歴管理を用いるトップダウン方式（英: top-down with memoization） – 分割統治法において、計算結果を記録（メモ化）して再利用する方法。再帰を併用する場合はメモ化再帰（英: memoized recursion）とも呼ばれる。
- ボトムアップ方式（英: bottom-up method） – 先に部分問題を解いていく方法

動的計画法の例：フィボナッチ数列

n番目のフィボナッチ数（0, 1, 1, 2, 3, 5, 8, 13,,,）をF_nで表すと、F_nは再帰的に以下の様に定義されます。

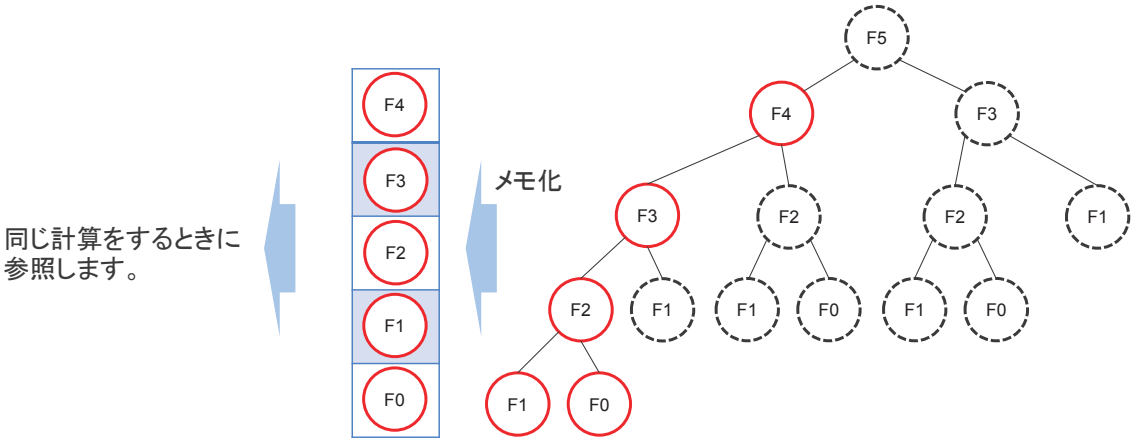
$$F_0 = 0,$$
$$F_1 = 1,$$
$$F_{n+2} = F_n + F_{n+1} (n \geqq 0)$$

F_5を計算するためには、F_4とF_3の値を計算する必要があります。
そしてF_4を計算するためにはF_3とF_2を計算する必要があります。
これをグラフにすると、F_0、F_1、F_2などは何度も登場することがわかります。
このように、nが大きくなるにしたがい、何度も同じ計算をする必要があります。



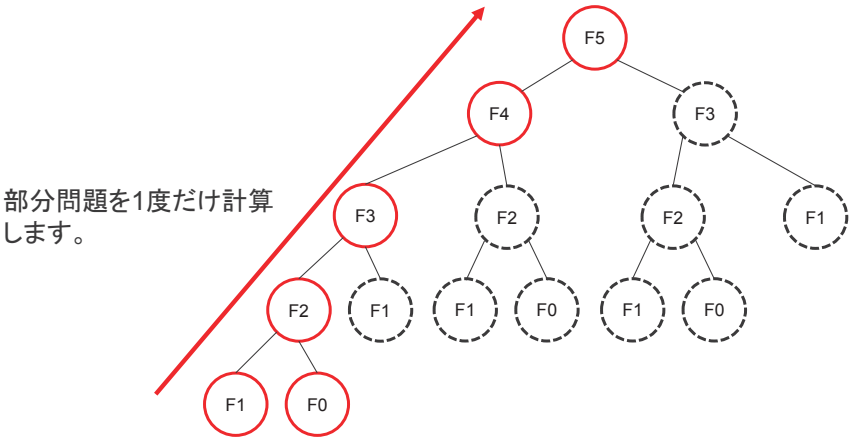
動的計画法の実現方法：履歴管理を用いるトップダウン方式

「一度計算した値をメモリなどに保持しておき、同じ計算をする必要がある場合はメモリに格納した値を参照する」というようにすると、計算量を削減することができます。



動的計画法の実現方法：ボトムアップ方式

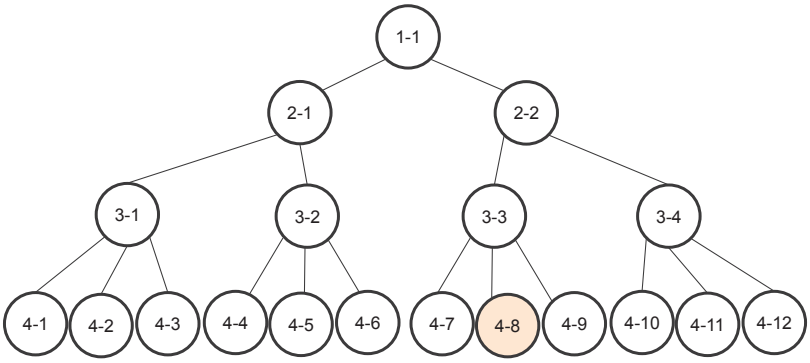
再帰処理が何度も発生するのは、トップダウンで上から下に向かって計算することが原因です。
逆に下から上に計算すれば1方向で何度も計算しなくて済む、という方法をボトムアップ方式といいます。



演習

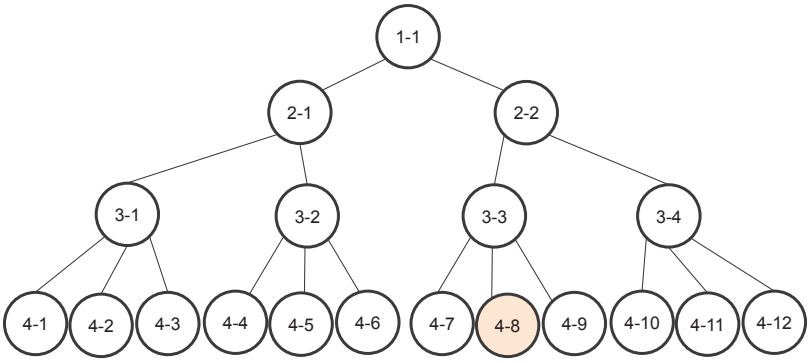
演習1：深さ優先探索

深さ優先探索する場合、[4-8]は何個目のノードとして参照されるでしょうか。
ただし、左側から先に探索されるとします。



演習2：幅優先探索

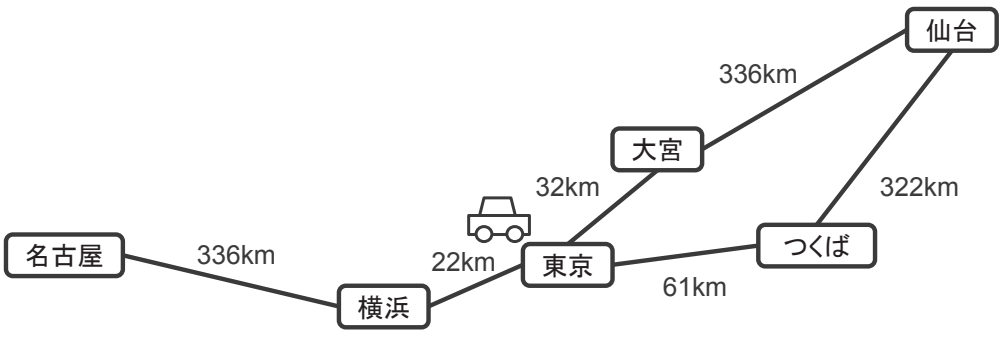
幅優先探索する場合、[4-8]は何個目のノードとして参照されるでしょうか。
ただし、左側から先に探索されるとします。



演習3：A*（エースター）探索

講義ではA*探索の例としてカーナビを扱いました。
その他にA*探索が適していると考えられる事例を挙げてください。

また、推定コスト$h(n)$を計算するのに最適な関数も考えてください。



第11回：遺伝的アルゴリズム

アジェンダ

- 遺伝的アルゴリズム
- 巡回セールスマン問題

遺伝的アルゴリズム

遺伝的アルゴリズムとは

[Wikipediaより]
遺伝的アルゴリズム（いでんてきアルゴリズム、英語：genetic algorithm、略称：GA）とは、1975年にミシガン大学のジョン・H・ホランド（John Henry Holland）によって提案された近似解を探索するメタヒューリスティックアルゴリズムである。人工生命同様、偶然の要素でコンピューターの制御を左右する。

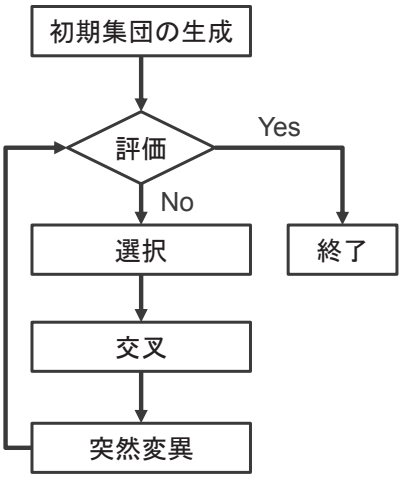
遺伝的アルゴリズムはデータ（解の候補）を遺伝子で表現した「個体」を複数用意し、適応度の高い個体を優先的に選択して交叉・突然変異などの操作を繰り返しながら解を探索する。

※ヒューリスティクスとは実験的手法によって答えを導出する方法で、個別の問題に特化した方法のことです。メタヒューリスティクスとは、同様に実験的手法ですが、個別の問題だけでなく様々な問題に応用できる汎用的な近似アルゴリズムのことです。

遺伝的アルゴリズムの処理フロー

遺伝的アルゴリズムを構成している重要な処理プロセスとして、以下の3つがあります。

- 選択 (selection)
- 交叉 (crossover)
- 突然変異 (mutation)

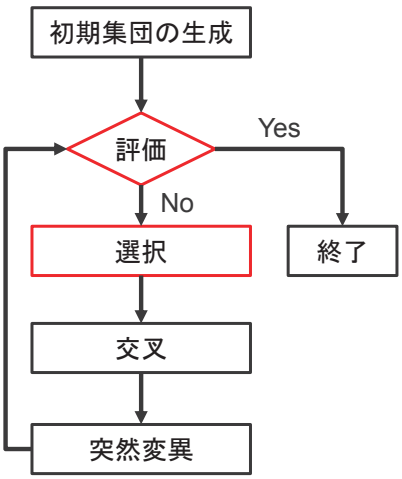


遺伝的アルゴリズムの処理フロー：選択

選択の指標となる適応度を評価します。
適応度はどれだけその個体が優れているかを示したもので、
値が大きいほど優れているとみなします。

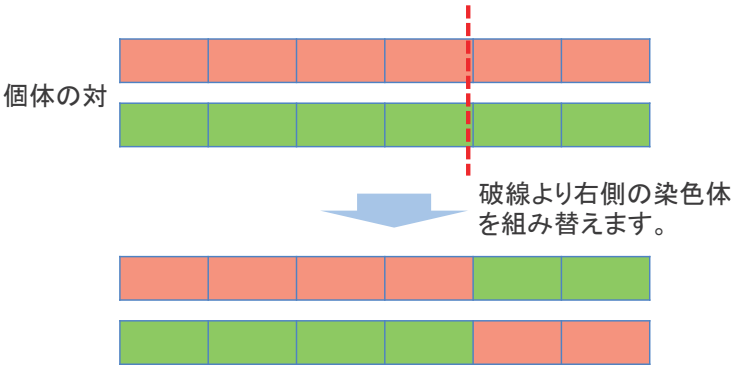
適応度を元に優秀な個体を選択します。
選択の方法として、以下のような方法があります。

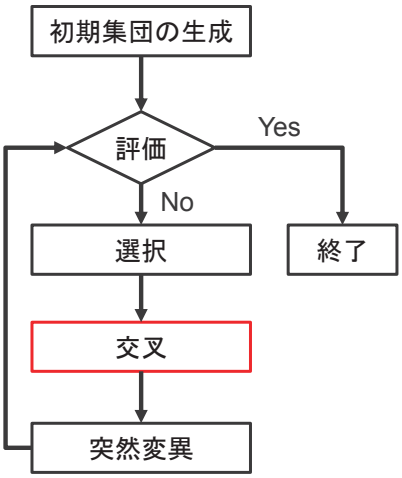
- 適応度比例戦略
  個体をランダムに選択する際に、適応度によって選ばれる確率が調整されます。
- エリート保存戦略
  最も適応度の高い固体をそのまま次世代に複製する方法です。
- トーナメント戦略
  ランダムに複数の個体を選び、最も適応度の高いものを親として残します。



遺伝的アルゴリズムの処理フロー：交叉

選択された個体間で対を作り、染色体の組み換えを行います。

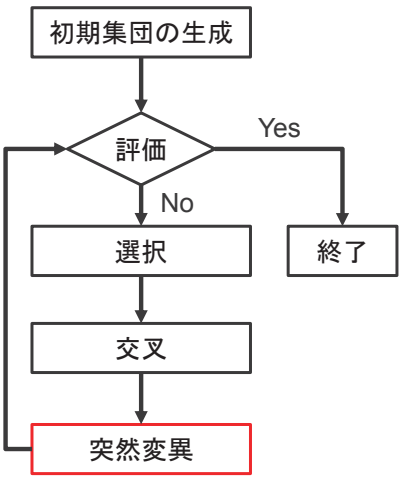




遺伝的アルゴリズムの処理フロー：突然変異

染色体に突然変異（ある確率で染色体の一部の値を変える操作）を加えます。

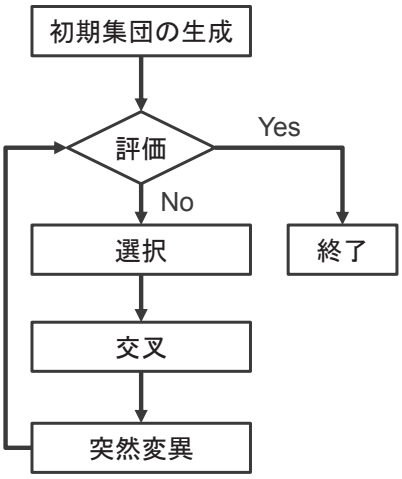
交叉だけでは個体の親に依存するような子しか生成されません。
突然変異させることで、交叉だけでは生成できない子を生成して
個体群の多様性を維持します。



遺伝的アルゴリズムの特徴

解空間構造が不明であり（決定的な優れた解法が発見されていない）、様々な可能性を実験的に全探索する必要があるような問題が存在します。
しかし、コンピュータリソースや計算時間を考えると、全探索は
適切な解放ではありません。

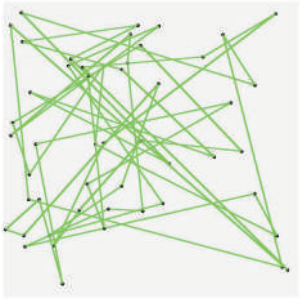
これまで学習してきた操作をみてわかるように、遺伝的アルゴリズムは
このような問題に対して有効です。



巡回セールスマン問題

巡回セールスマン問題

あるセールスマンが、いくつかの地点を訪問して、スタート地点に戻ってくるとします。
各地点間の距離は分かっているとしたとき、なるべく総移動距離が短くなるように各地点を回る順番を決める、という問題が巡回セールスマン問題です。



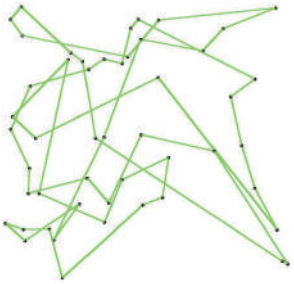


図2　第1世代から第200世代までの最短経路の変遷 (elite_rate = 0.1, mutation_rate = 0.03)　　図3　10000世代計算後の最短経路 (elite_rate = 0.1, mutation_rate = 0.03)

参照：http://blog.serverworks.co.jp/tech/2016/06/27/genetic_algorithm_traveling_salesman_ec2/

巡回セールスマン問題の計算コスト

巡回セールス問題は、地点数が増えると計算をコンピュータに実行させとしても「全探索して総移動距離が最短になる順番を見つける」という方法は現実的ではなくなります。

順番のパターン数は、じゅず順列と同じ考え方で計算でき、地点数がNのとき地点を訪問する順番は(N − 1)!/2 通りになります。

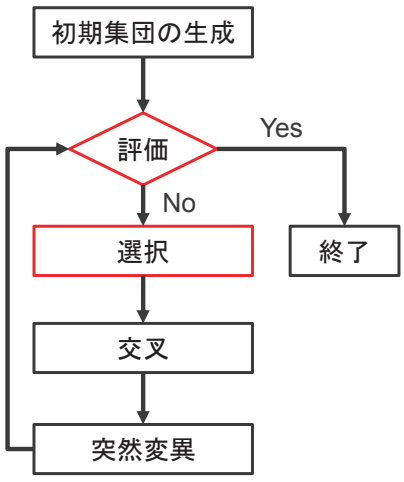
地点数：5	->	訪問する順番のパターン数：12
地点数：10	->	訪問する順番のパターン数：約18万
地点数：15	->	訪問する順番のパターン数：約400億

巡回セールスマン問題のような課題を解決するには、「一部を調べてみて、そこそこ移動距離が短くなる順番を見つける」という方法を取らざるを得ません。このようなときに、遺伝的アルゴリズムが応用できます。

演習

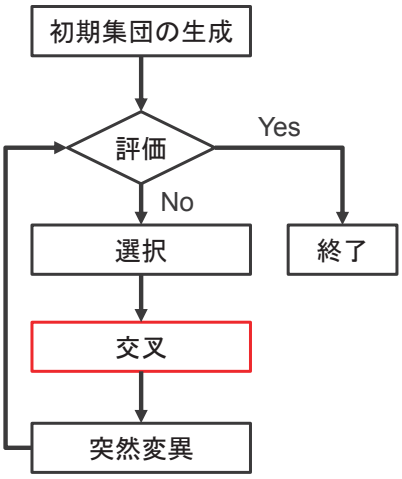
演習1：遺伝的アルゴリズム（選択）

遺伝的アルゴリズムにおける「選択」とはどのような操作を行うのか、説明してください。



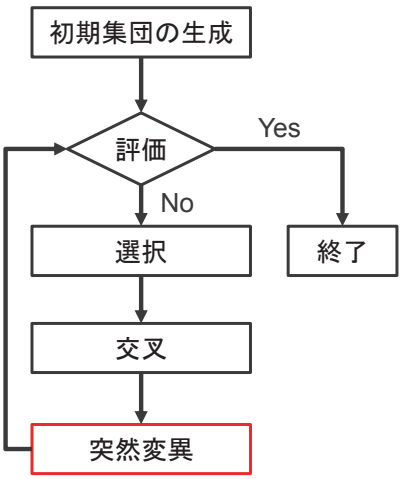
演習2：遺伝的アルゴリズム（交叉）

遺伝的アルゴリズムにおける「交叉」とはどのような操作を行うのか、説明してください。



演習3：遺伝的アルゴリズム（突然変異）

遺伝的アルゴリズムにおける「突然変異」とはどのような操作を行うのか、説明してください。



演習4：巡回セールスマン問題

巡回セールスマン問題を遺伝的アルゴリズムで解くとします。
適応度を評価するための評価関数として、どのような関数が適切でしょうか。
数式で表現してください。

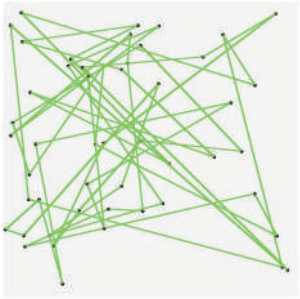


図2　第1世代から第200世代までの最短経路の変遷 (elite_rate = 0.1, mutation_rate = 0.03)

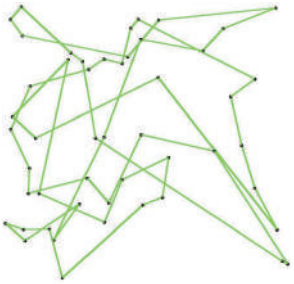


図3　10000世代計算後の最短経路 (elite_rate = 0.1, mutation_rate = 0.03)

参照：http://blog.serverworks.co.jp/tech/2016/06/27/genetic_algorithm_traveling_salesman_ec2/

第12回：人工知能概論総復習その2

第7回：ゲーム理論

ゲーム理論とは

[Wikipediaより]
ゲーム理論（ゲームりろん、英: game theory）とは、社会や自然界における複数主体が関わる意思決定の問題や行動の相互依存的状況を数学的なモデルを用いて研究する学問である。数学者ジョン・フォン・ノイマンと経済学者オスカー・モルゲンシュテルンの共著書『ゲームの理論と経済行動』(1944年) によって誕生した。元来は主流派経済学（新古典派経済学）への批判を目的として生まれた理論であったが、1980年代の「ゲーム理論による経済学の静かな革命」を経て、現代では経済学の中心的役割を担うようになった。

ゲーム理論の対象はあらゆる戦略的状況（英: strategic situations）である。「戦略的状況」とは自分の利得が自分の行動の他、他者の行動にも依存する状況を意味し、経済学で扱う状況の中でも完全競争市場や独占市場を除くほとんどすべてはこれに該当する。さらにこの戦略的状況は経済学だけでなく経営学、政治学、法学、社会学、人類学、心理学、生物学、工学、コンピュータ科学などのさまざまな学問分野にも見られるため、ゲーム理論はこれらにも応用されている。

囚人のジレンマ

2人の囚人AとBが逮捕されました。お互い独房に入れられ、意思疎通ができないとします。
二人に与えられた選択肢は「自白する」「自白しない」の二つです。
その際、警察はAとBそれぞれに以下のような条件を提示しました。

・AとBが両方自白しないなら、両方懲役1年となる。
・片方だけが自白した場合、自白した方は懲役なし、しなかった方は懲役5年となる。
・両方自白した場合、両方とも懲役3年となる。

2人の囚人AとBはどのような行動を取るでしょうか？

		囚人A	
		自白する	自白しない
囚人B	自白する	A：懲役3年 B：懲役3年	A：懲役5年 B：釈放
	自白しない	A：釈放 B：懲役5年	A：懲役1年 B：懲役1年

パレート最適

パレート最適とは、全体として最も利益が大きく最適な状態のことです。
囚人のジレンマにおいては、両者とも自白せず懲役1年ずつになる場合です。

囚人個人にとっては、「これ以上に利益を得ようと思ったら、誰か他の人が不利益を被る必要がある状態」とも言えます。
つまり、囚人AもしくはB個人の視点から考えると、懲役1年より利益を得る状態は、釈放されることです。
これが成り立つには、自分だけ自白し、相手は自白しないことが必要で、自白しなかった相手は懲役5年となってしまいます。

		囚人A	
		自白する	自白しない
囚人B	自白する	A：懲役3年 B：懲役3年	A：懲役5年 B：釈放
	自白しない	A：釈放 B：懲役5年	A：懲役1年 B：懲役1年

ナッシュ均衡

ナッシュ均衡とは、個々の最適を考えて合理的な判断をした結果ある均衡点に落ち着き、両者とも自分だけがそこから動く利益のない状態のことです。

囚人のジレンマの問題では、各々にとって最高の結果は自分の懲役が0年になることであり、最悪の結果は自分の懲役が5年になることです。
全体の最適化（パレート最適）を考えるとお互いに懲役1年になることがベストですが、2人は独房に入れられているため、話し合ってそれを到達する、というのは困難でしょう。
自分だけが懲役5年になるリスクを負わない合理的な行動（囚人AとBともに自白する）を取る可能性があります。

		囚人A	
		自白する	自白しない
囚人B	自白する	A：懲役3年 B：懲役3年	A：懲役5年 B：釈放
	自白しない	A：釈放 B：懲役5年	A：懲役1年 B：懲役1年

支配戦略

支配戦略とは、他のプレイヤーがどのような決断をしたとしても、自分のとるべき行動が決まってる状態のことです。
相手の選択に関わらず自分の行動が最適化されるため、選択を変える必要が無くなります。

囚人のジレンマの場合、相手が自白しようがしまいが、自分は自白をしておけば最も大きなリスクである懲役5年を避けられます。囚人のジレンマだと、両者ともに支配戦略を選択できる状況が生まれます。

お互いに支配戦略ができる状態を「支配戦略均衡」といい、支配戦略均衡はお互いに自らの選択を変える必要がなくなるため、ナッシュ均衡と同じ意味を持ちます。

		囚人A	
		自白する	自白しない
囚人B	自白する	A：懲役3年 B：懲役3年	A：懲役5年 B：釈放
	自白しない	A：釈放 B：懲役5年	A：懲役1年 B：懲役1年

社会的ジレンマ

自分にとって最も合理的だが周囲に非協力的な判断をした場合に最高の結果が得られる（相手は自白しないのに、自分だけ自白すると、自分は釈放され相手は懲役5年となる）。
しかし全員がその判断をした場合、全員で周囲に協力的な判断をした場合より悪い結果になる（2人とも自白すると、お互いに懲役3年になる）。

このような状況を社会的ジレンマといいます。

		囚人A	
		自白する	自白しない
囚人B	自白する	A：懲役3年 B：懲役3年	A：懲役5年 B：釈放
	自白しない	A：釈放 B：懲役5年	A：懲役1年 B：懲役1年

ゲーム理論の例：待ち合わせ

恋人と待ち合わせをしているとします。
その日は運悪く、二人ともスマートフォンを忘れてしまいました。
ですが、待ち合わせの時間に渋谷駅で待ち合わせ、という情報は覚えていたとします。

駅には多数の沿線が乗り入れていますし、改札も複数あります。連絡手段があれば、詳細に待ち合わせ場所をリアルタイムで話すことができるのですが、そうも行きません。

自分も相手も動き回ってなかなか会えない、ということになりかねません。
この時下手に二人とも動くよりは、片方だけが動いてもう片方を探し回る方が早く会える確率は高いです。しかし二人とも動かないという状況は最悪なため、二人ともそれを恐れ動き回ってしまいます。
この場合、片方だけが動いている状態がパレート最適、二人とも動いている状態がナッシュ均衡です。

ゲーム理論の例：偏差値

定期テストや入試などの成績は相対評価で行われます。
みんなが勉強しなくて点数が総じて低ければ、自分の点数が低くても相対評価は高くなります。
反対に、みんなが一生懸命勉強すれば、自分も必死に頑張らなければ評価が下がります。

みんながどれくらい頑張っているのだろう？と心配になったり、自分は全く勉強していないふりをして、隠れて必死に勉強するなど、駆け引きが生じてしまいます。

ゲーム理論の例：ホテルの宿泊料金設定

ある地域で人気グループのコンサートが開催されます。
地方から駆けつけるお客さんも多く、コンサート会場周辺の宿泊施設は、予約がいっぱいになる見込みです。

宿泊施設の協会に所属しているオーナーたちは、「お互いに宿泊料金を下げないようにしよう」と約束したものの、協会に所属していないホテルが価格を下げれば、先ずはそのホテルにお客が流れます。協会に所属しているオーナーたちは、コンサートの日が近づくに連れて、自分のホテルの空室率が気になって仕方ありません。

AIを駆使してコンサートチケットやホテル宿泊料金などを調整し、施設全体で収益を最大化しよう、という考え方があり、レベニューマネジメントと言われています。
ライバルホテルの宿泊料金、予約率がターゲットの日（コンサート日）に近づくにつれて変動する中で、自分のホテルの宿泊料金をどのように設定するのか、非常に面白い分野です。

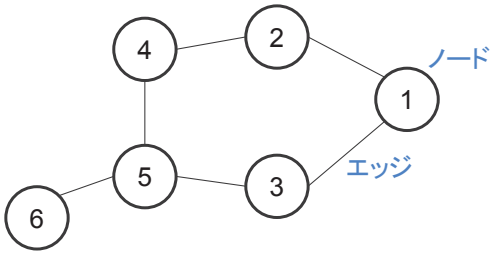
演習：ゲーム理論の事例

皆さんの身近にあるゲーム理論の事例を挙げてください。

第8回：グラフ理論その1

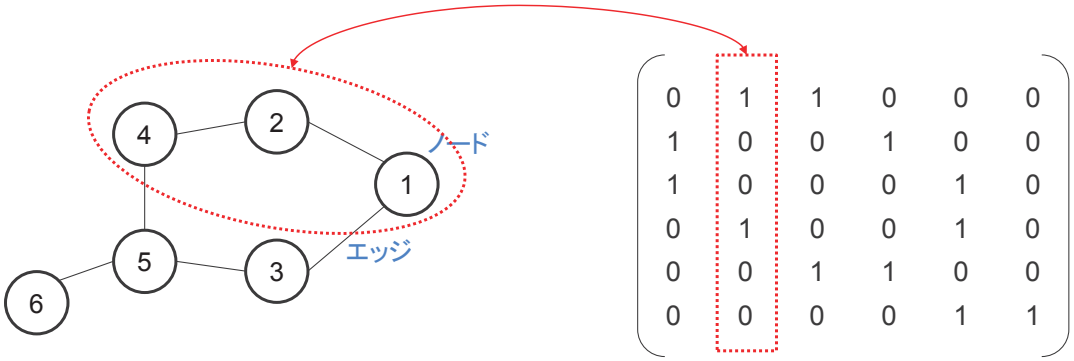
グラフ理論とは

グラフ理論（ Graph theory）とは、ノード（節点・頂点）の集合とエッジ（枝・辺）の集合で構成されるグラフに関する数学の理論です。



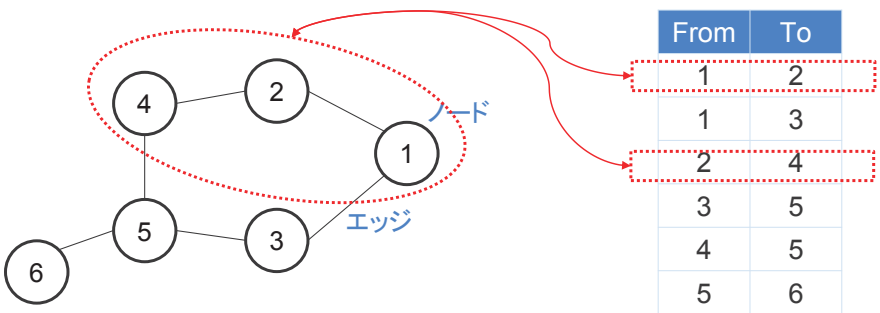
グラフ理論の表現方法

グラフの表現には、図ではなく行列を使う方法もあります。
下の例では、ノードの数6を辺とする6×6の行列で表現しています。



グラフ理論の表現方法

グラフを行列で表現した場合、ノードの数がNだとすると、必ずN^2の情報を保持することになります。
これでは、分析の際にコンピュータのリソースが枯渇してしまいます。
つながり（エッジ）が存在しないノードは無視し、つながりが存在するノードのみの情報で表現する方法もあります。

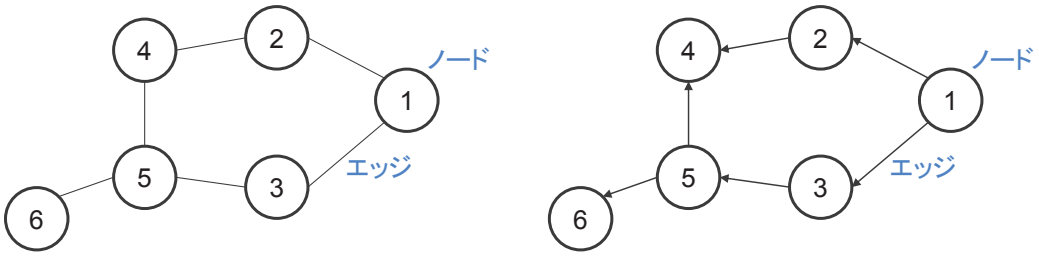


From	To
1	2
1	3
2	4
3	5
4	5
5	6

※From / Toと記載されていますが、エッジには方向がない無向グラフを例にしているため、[From:1, To:2]と[From:2, To:1]は同じ意味になります。

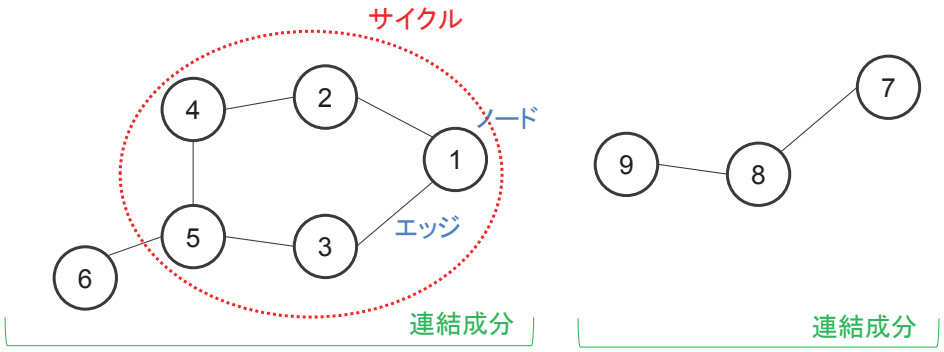
無向グラフと有向グラフ

エッジに向きがないグラフ(下左図)を無向グラフといいます。
エッジに向きがあるグラフ(下右図)を有向グラフといいます。



サイクル、連結成分

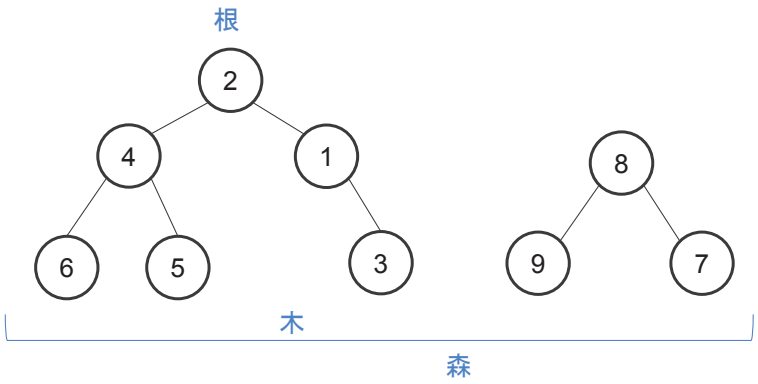
ノードとエッジを辿ると元に戻ってくる経路をサイクルといいます。
パスがつながっているかたまりを連結成分といいます。



木

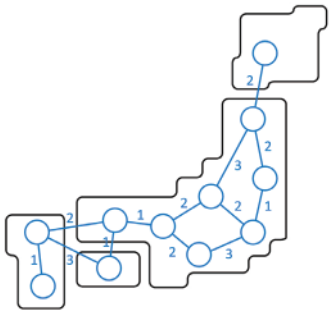
連結でサイクルがないグラフを木といいます。
木が複数あるものを森といいます。

木は、あるノードを根として、そこからグラフがはじまると考えることがあります。

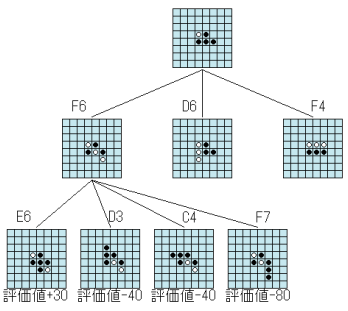


グラフ探索とは

ある状態や地点をノードで表現し、状態の遷移や地点間の移動をエッジで表現し、目的とする状態までのコストを最小とする経路を探すことを、グラフ探索といいます。



※参照（最短経路）：https://tajimarobotics.com/graph-theory-algorithm/

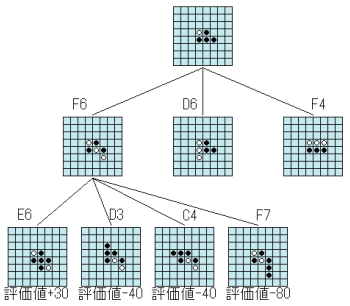


※参照（ゲーム木）：http://www.es-cube.net/es-cube/reversi/sample/html/2_3.html

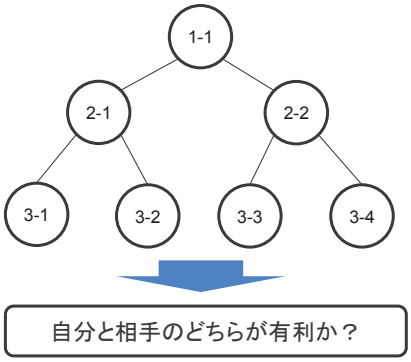
ミニマックス法

オセロを例としてミニマックス法を学習します。

オセロのように、自分にとっては最も有利な手を自分が打ち（max）、次に相手が自分にとって最も不利な手を打ち（min）、それらが交互に繰り返されるゲームの打ち手を考える際にゲーム木というものを利用します。

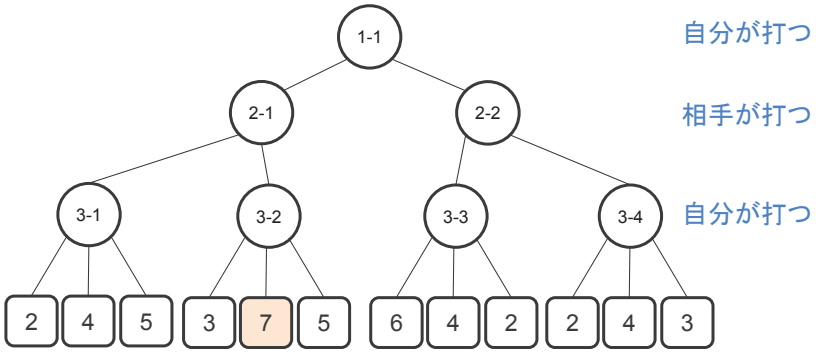


※参照（ゲーム木）：http://www.es-cube.net/es-cube/reversi/sample/html/2_3.html



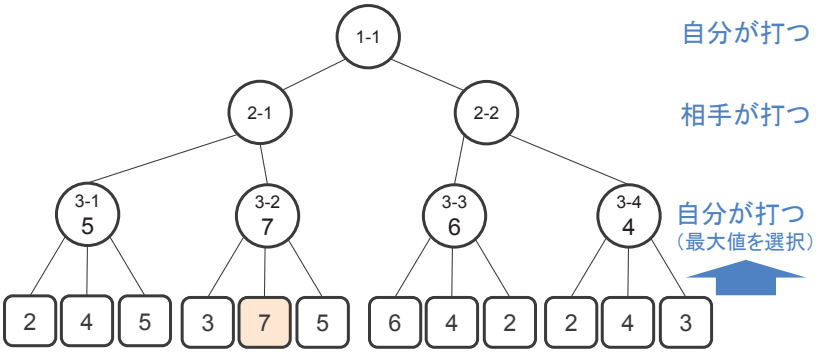
ミニマックス法

3手先までを考えてみます。
3手先までの全てのケースを想定してスコアリングした結果、自分が7の状態にあることが最良だとわかりました。スコア7にたどり着けるでしょうか？



ミニマックス法

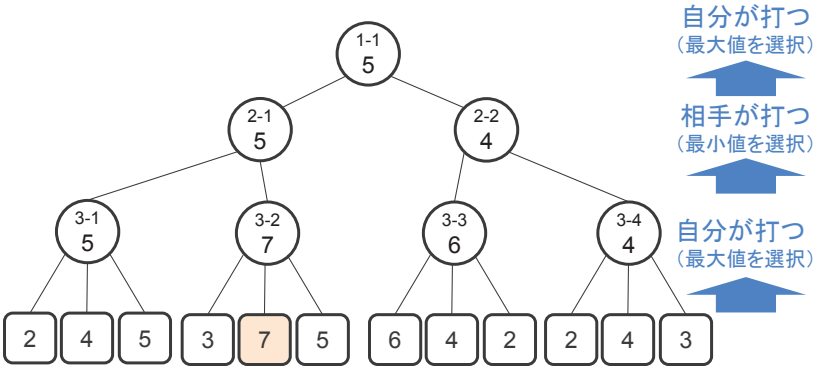
一番下から上に上がっていきます。
まず、3手目の自分の番では、スコアが最も大きくなるケースを選びます。



ミニマックス法

次に、2手目の相手の番では、スコアが最も低くなるケースが選ばれると想定します。
1手目の自分の番では、スコアが最も高くなるケースを選びます。

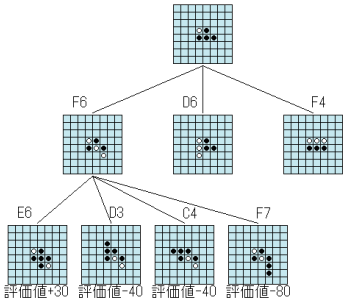
最終的に、自分が得られうる最良の状態は、スコア5で有ることがわかりました。



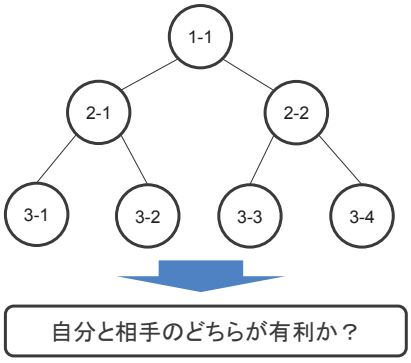
ネガマックス法

オセロを例としてネガマックス法を学習します。

ミニマックス法との違いは、相手はこちらの利益を最小にするように打つのではなく、相手自身の利益を最大にするように打つというところです。



※参照（ゲーム木）：http://www.es-cube.net/es-cube/reversi/sample/html/2_3.html

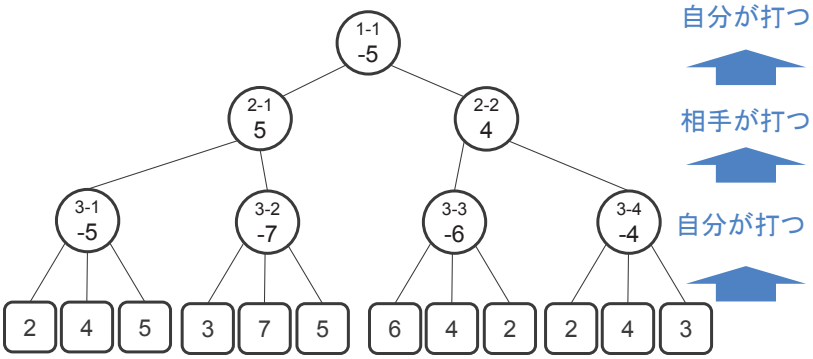


ネガマックス法

一番下の3手目から考えます。
自分に最も良い状態は、左から[5, 7, 6, 4]の状態になることです。
相手にとっては、-1を掛けた[-5, -7, -6, -4]となります。

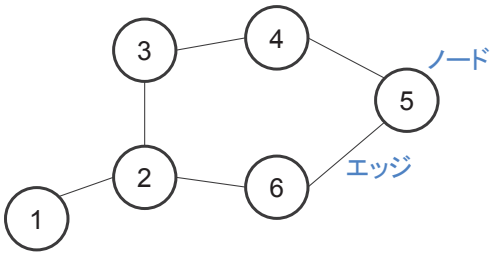
相手は、相手のスコアが最大になるように行動すると仮定しますので、[-5, -4]を選択します。
自分にとっては-1を掛けた[5, 4]となります。

最終的に5を選択するので、自分にとっては
最良の状態はスコア5となります。



演習1：グラフ理論の表現方法

左下に図で表現しているグラフを、From-Toの形式で表現してください。
ただし、無向グラフとします。

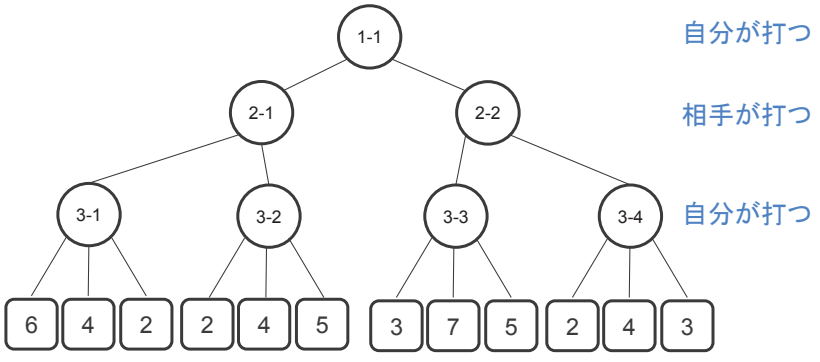


From	To
?	?
?	?
?	?
?	?
?	?
?	?

※From / Toと記載されていますが、エッジには方向がない無向グラフを例にしているため、[From:1, To:2]と[From:2, To:1]は同じ意味になります。

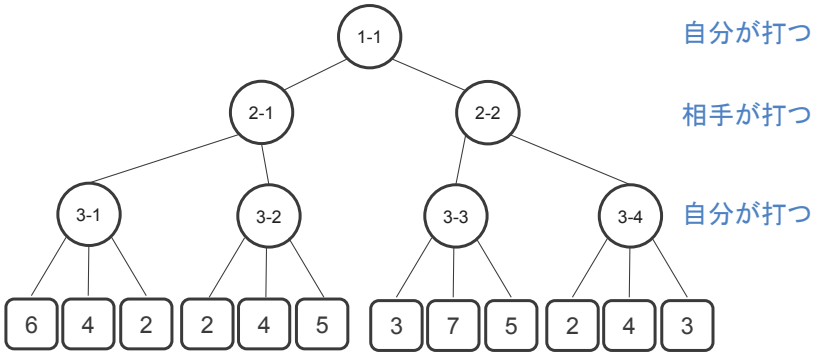
演習2：ミニマックス法

オセロの3手先を考えます。
ミニマックス法を使って、自分が得られるであろう最良のスコアを求めてください。



演習2：ネガマックス法

オセロの3手先を考えます。
ネガマックス法を使って、自分が得られるであろう最良のスコアを求めてください。

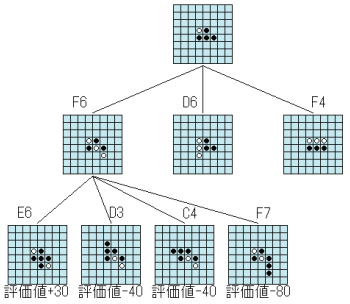


第9回：グラフ理論その2

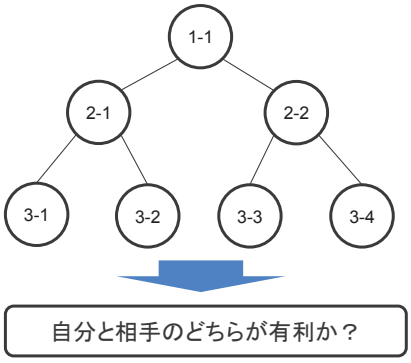
アルファ-ベータ法

アルファ-ベータ法は、ミニマックス法を改良したアルゴリズムです。

全ての手を探索すると膨大な時間を要するため、探索ノード数を削減するように工夫したアルゴリズムです。



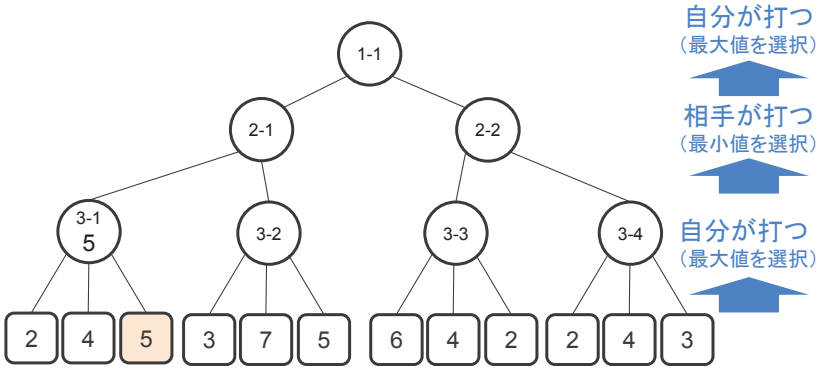
※参照（ゲーム木）：http://www.es-cube.net/es-cube/reversi/sample/html/2_3.html



アルファ-ベータ法

各ノードで、必ず左側から探索するとします。

まず[1-1] -> [2-1] -> [3-1]と探索します。
[3-1]に属している数字で最も大きな数字は5です。

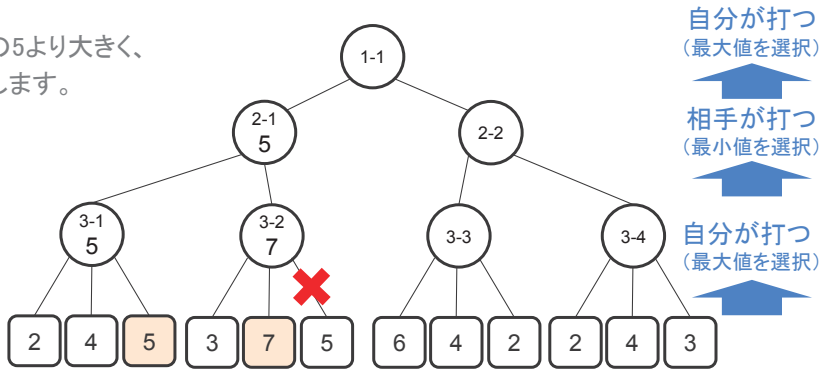


アルファ-ベータ法

次に[1-1] -> [2-1] -> [3-2]と探索します。
[3-2]に属している数字を左から見ていきます。

ミニマックス法では、[3-2]において3, 7, 5から最大の数字を選び、
[2-1]においては[3-1]と[3-2]で小さい数字を選びます。

[3-2]において7が見つかった時点で、[3-1]の5より大きく、
[2-1]では[3-1]の5が選択されることが決定します。
よって[3-2]において5の探索はしません。



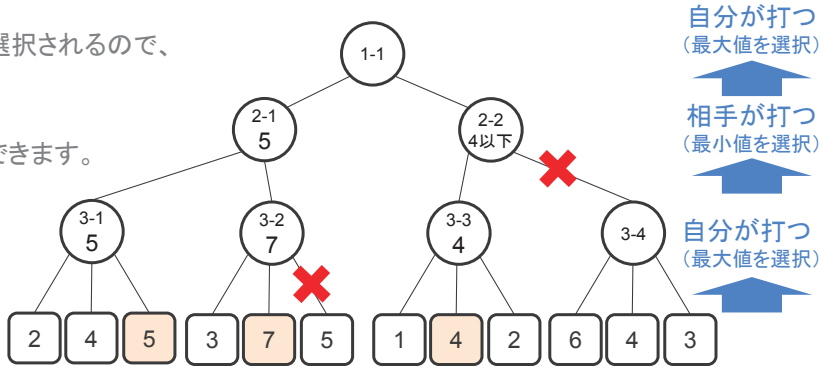
アルファ-ベータ法

次に[1-1] -> [2-2] -> [3-3]と探索します。
[3-3]において最大値の4が選択されます。

[2-2]は、[3-3]と[3-4]の小さい値が選択されます。
[3-3]において4となったということは、[2-2]は4以下であるということです。

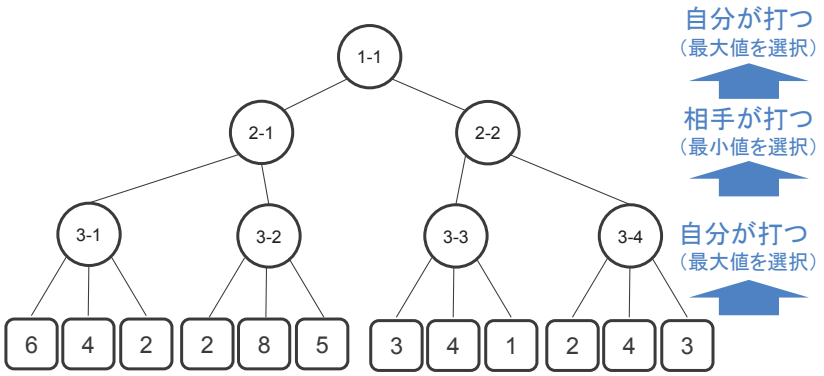
[1-1]においては[2-1]と[2-2]の大きい値が選択されるので、
[2-1]の5が選択されます。

よって、[3-4]以下の探索は省略することができます。



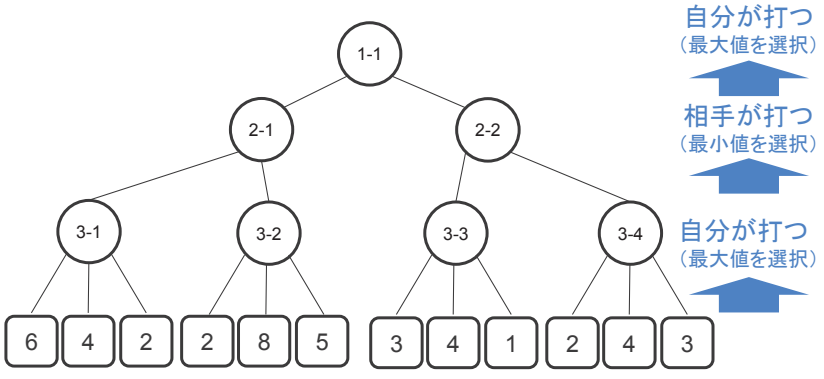
演習1：ミニマックス法

オセロの3手先を考えます。
ミニマックス法を使って、自分が得られるであろう最良のスコアを求めてください。



演習2：アルファ-ベータ法

オセロの3手先を考えます。
アルファ-ベータ法を使って、自分が得られるであろう最良のスコアを求めてください。



第10回：グラフ理論その3

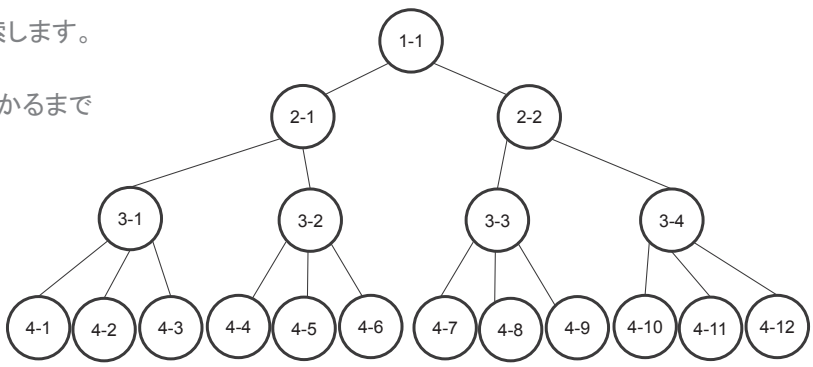
深さ優先探索

深さ優先探索とは、先ずは行けるところまでまっすぐ探索し、行き止まりになったら1つ手前に戻り別のエッジを探索するという手法です。

左側から探索するとすると、[1-1]->[2-1]->[3-1]->[4-1]と進みます。
[4-1]で行き止まりになるので[3-1]に戻り、[4-2][4-3]を探索します。

次に[2-1]まで戻り、[2-1]->[3-2]->[4-4]と探索します。

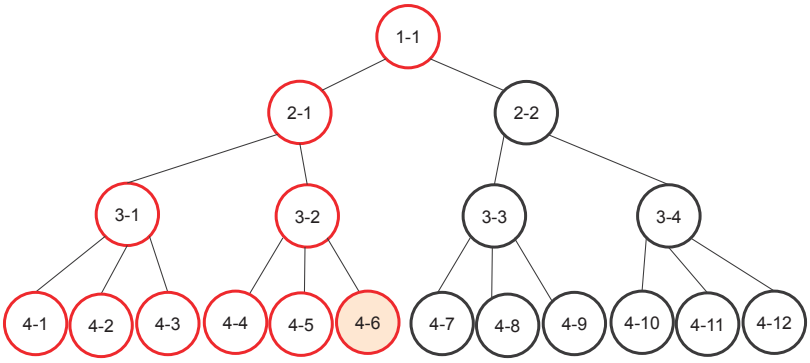
このような探索の仕方を、目的のノードが見つかるまで繰り返します。



深さ優先探索の計算量

深さ優先探索の計算量を考えてみます。
[4-6]を見つけるまで、何個のノードを探索する必要があるでしょうか。

左側から探索するとした場合、[4-6]は10個目のノードとして見つかります。

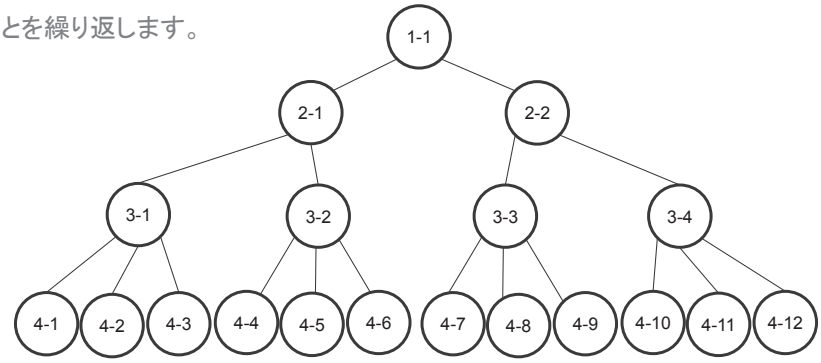


幅優先探索

幅優先探索とは、出発点に近い点から探索するという手法です。

左側から探索するとすると、[1-1]に隣接している[2-1]->[2-2]と進みます。
次に、[2-1]に隣接している [3-1][3-2]を探索します。
次に、[2-2]に隣接している [3-3][3-4]を探索します。

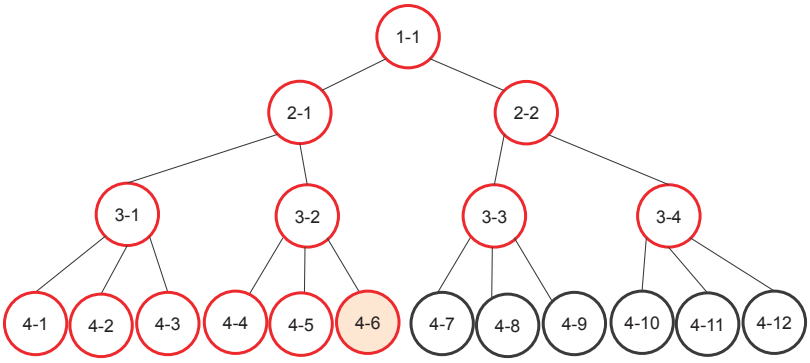
このような出発点から横方向に探索していくことを繰り返します。



幅優先探索の計算量

幅優先探索の計算量を考えてみます。
[4-6]を見つけるまで、何個のノードを探索する必要があるでしょうか。

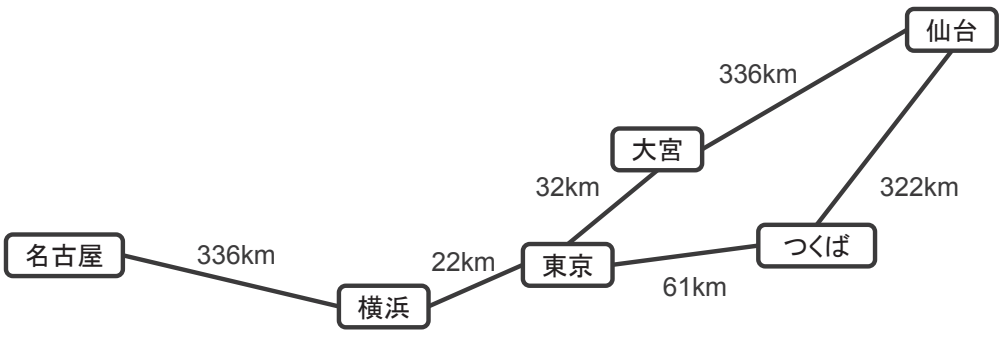
左側から探索するとした場合、[4-6]は13個目のノードとして見つかります。



A*（エースター）探索

[Wikipediaより]
A* アルゴリズムは、「グラフ上でスタートからゴールまでの道を見つける」というグラフ探索問題において、ヒューリスティック関数 h(n) という探索の道標となる関数を用いて探索を行うアルゴリズムである。

例えば、東京から仙台まで旅行に行くことを考えます。
どのルートを辿ればよいでしょうか？



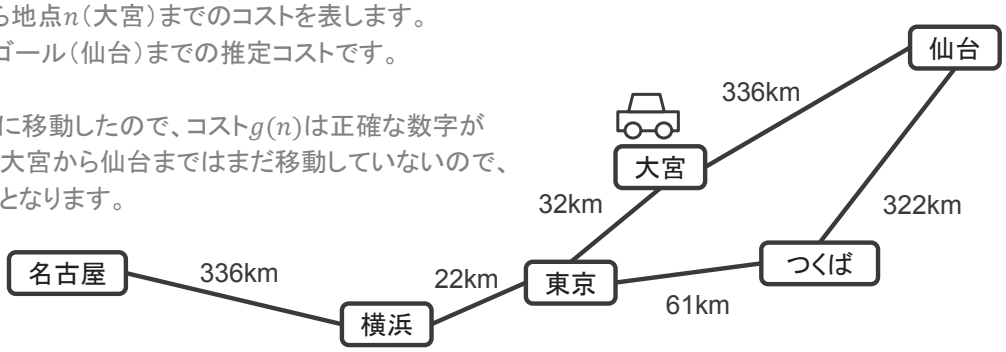
A*（エースター）探索

A*探索では、コスト関数$f(n)$を考えます。
もし最短距離で到着したい人は、コスト関数は最短距離を返す関数となります。
距離でなく時間を優先したい人は、最短時間を返す関数となります。

$$f(n) = g(n) + h(n)$$

例えば東京を出発し仙台を目指すとします。現在は大宮にいるとします。
$g(n)$はスタート（東京）から地点n（大宮）までのコストを表します。
$h(n)$は地点n（大宮）からゴール（仙台）までの推定コストです。

※東京から大宮まで実際に移動したので、コスト$g(n)$は正確な数字がわかっています。ですが、大宮から仙台まではまだ移動していないので、$h(n)$はあくまで推定コストとなります。



A*（エースター）探索

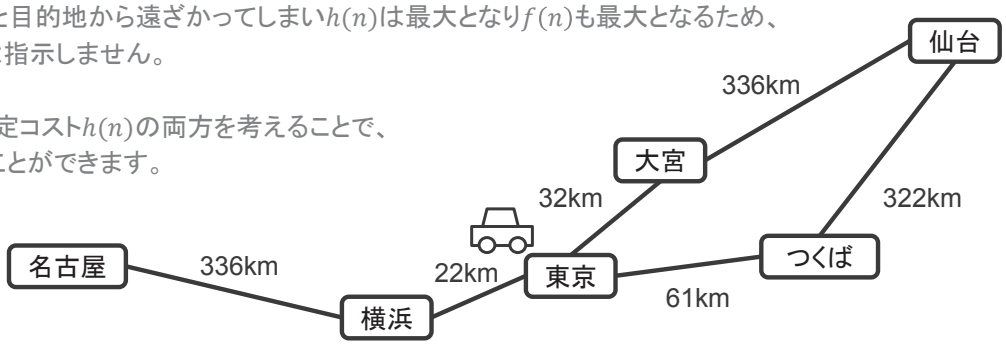
A*探索で、どのように経路を選択するのか見てみましょう。コスト関数は最短距離を返す関数とします。

$$f(n) = g(n) + h(n)$$

また、推定コスト$h(n)$は各都市の緯度経度から計算したユークリッド距離$\sqrt{x^2 + y^2}$を考えるとします。

スタートの東京にいるあなたが、横浜/大宮/つくばに移動した3つのパターンを考えてみます。
$g(n)$のみを考慮すれば、東京から横浜に移動したパターンが、コストは最小（22km）になります。
ですが、横浜に移動すると目的地から遠ざかってしまい$h(n)$は最大となり$f(n)$も最大となるため、
ナビは横浜に行くようには指示しません。

このようにコスト$g(n)$と推定コスト$h(n)$の両方を考えることで、
最適なルートを検索することができます。



動的計画法とは

[Wikipediaより]
動的計画法（どうてきけいかくほう、英: Dynamic Programming, DP）は、計算機科学の分野において、アルゴリズムの分類の1つである。対象となる問題を複数の部分問題に分割し、部分問題の計算結果を記録しながら解いていく手法を総称してこう呼ぶ。
細かくアルゴリズムが定義されているわけではなく、下記2条件を満たすアルゴリズムの総称である。

- 帰納的な関係の利用：より小さな問題例の解や計算結果を帰納的な関係を利用してより大きな問題例を解くのに使用する。
- 計算結果の記録：小さな問題例、計算結果から記録し、同じ計算を何度も行うことを避ける。帰納的な関係での参照を効率よく行うために、計算結果は整数、文字やその組みなどを見出しにして管理される。

以下の2種類の実現方法がある。

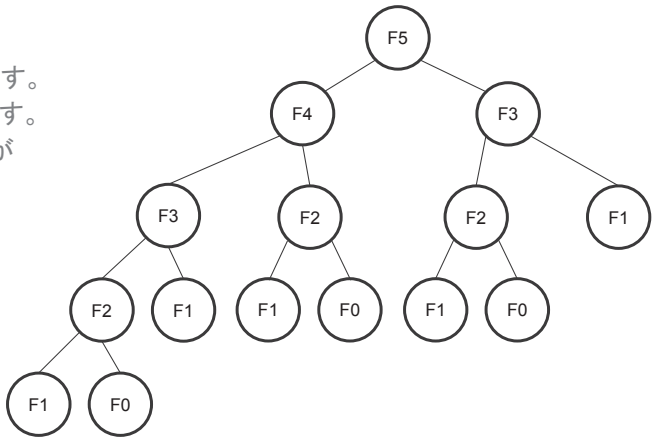
- 履歴管理を用いるトップダウン方式（英: top-down with memoization） – 分割統治法において、計算結果を記録（メモ化）して再利用する方法。再帰を併用する場合はメモ化再帰（英: memoized recursion）とも呼ばれる。
- ボトムアップ方式（英: bottom-up method） – 先に部分問題を解いていく方法

動的計画法の例：フィボナッチ数列

n番目のフィボナッチ数（0, 1, 1, 2, 3, 5, 8, 13,,,）をF_nで表すと、F_nは再帰的に以下の様に定義されます。

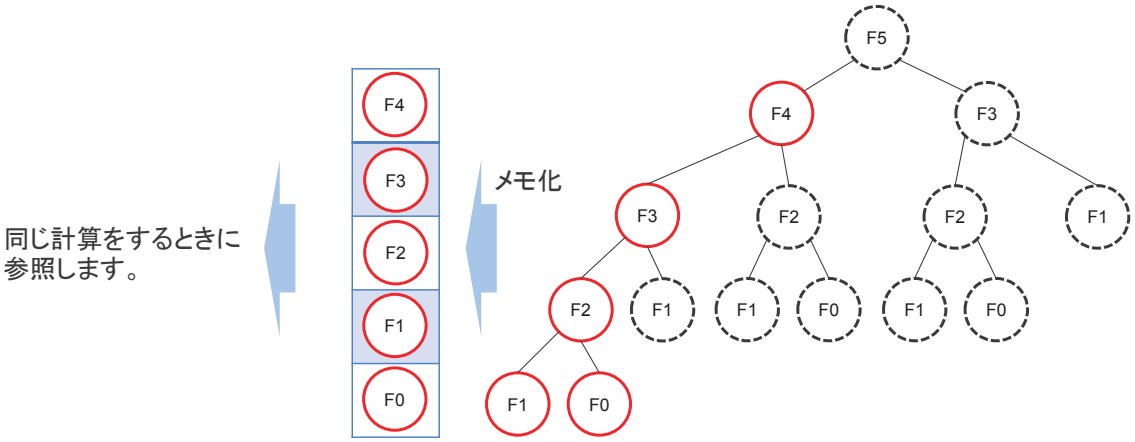
$$F_0 = 0,$$
$$F_1 = 1,$$
$$F_{n+2} = F_n + F_{n+1} (n \geqq 0)$$

F_5を計算するためには、F_4とF_3の値を計算する必要があります。
そしてF_4を計算するためにはF_3とF_2を計算する必要があります。
これをグラフにすると、F_0、F_1、F_2などは何度も登場することがわかります。
このように、nが大きくなるにしたがい、何度も同じ計算をする必要があります。



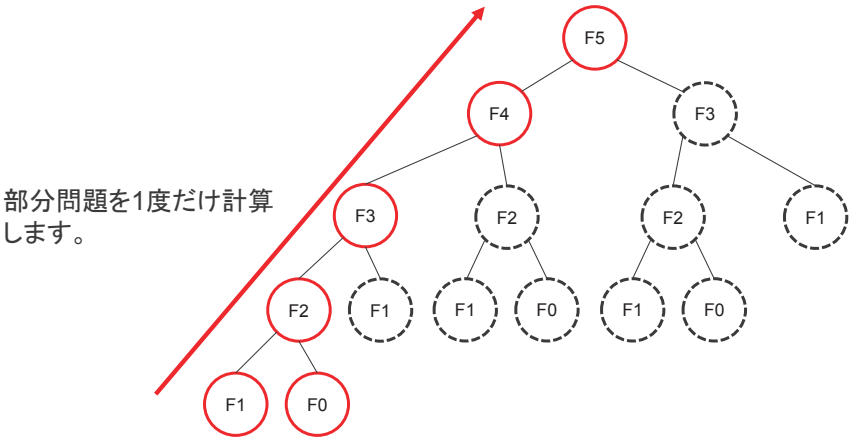
動的計画法の実現方法：履歴管理を用いるトップダウン方式

「一度計算した値をメモリなどに保持しておき、同じ計算をする必要がある場合はメモリに格納した値を参照する」というようにすると、計算量を削減することができます。



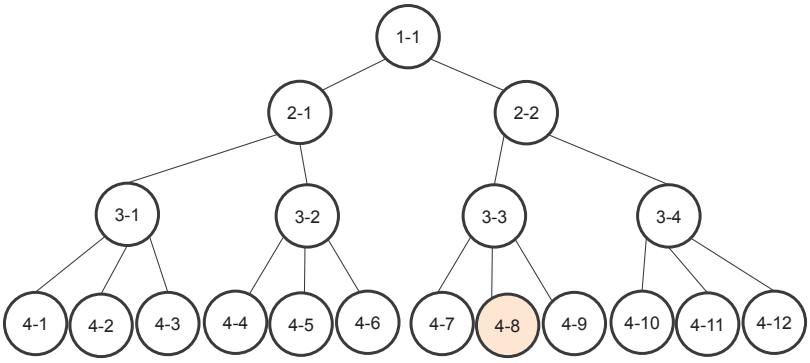
動的計画法の実現方法：ボトムアップ方式

再帰処理が何度も発生するのは、トップダウンで上から下に向かって計算することが原因です。
逆に下から上に計算すれば1方向で何度も計算しなくて済む、という方法をボトムアップ方式といいます。



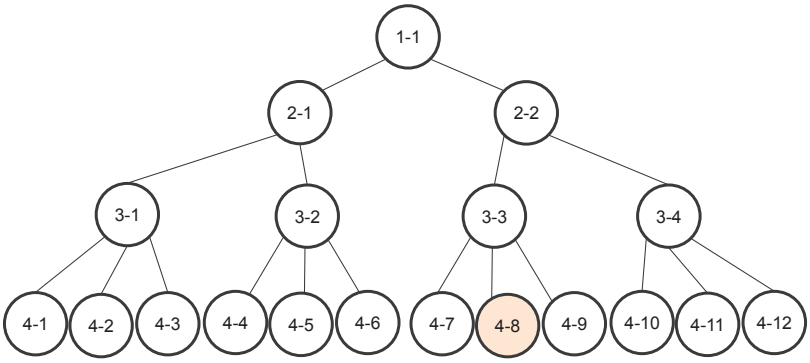
演習1：深さ優先探索

深さ優先探索する場合、[4-8]は何個目のノードとして参照されるでしょうか。
ただし、左側から先に探索されるとします。



演習2：幅優先探索

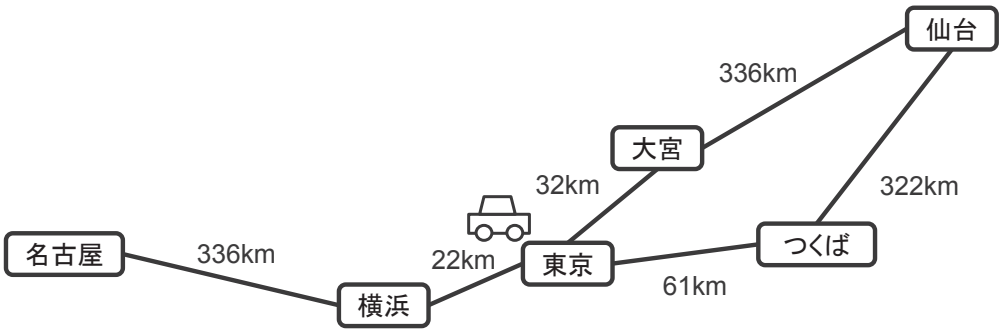
幅優先探索する場合、[4-8]は何個目のノードとして参照されるでしょうか。
ただし、左側から先に探索されるとします。



演習3：A*（エースター）探索

講義ではA*探索の例としてカーナビを扱いました。
その他にA*探索が適していると考えられる事例を挙げてください。

また、推定コスト$h(n)$を計算するのに最適な関数も考えてください。



第11回：遺伝的アルゴリズム

遺伝的アルゴリズムとは

[Wikipediaより]
遺伝的アルゴリズム（いでんてきアルゴリズム、英語：genetic algorithm、略称：GA）とは、1975年にミシガン大学のジョン・H・ホランド（John Henry Holland）によって提案された近似解を探索するメタヒューリスティックアルゴリズムである。人工生命同様、偶然の要素でコンピューターの制御を左右する。

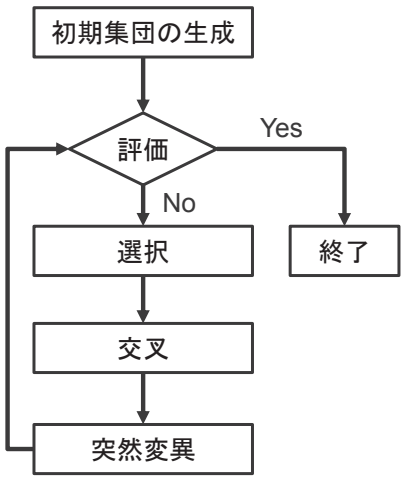
遺伝的アルゴリズムはデータ（解の候補）を遺伝子で表現した「個体」を複数用意し、適応度の高い個体を優先的に選択して交叉・突然変異などの操作を繰り返しながら解を探索する。

※ヒューリスティクスとは実験的手法によって答えを導出する方法で、個別の問題に特化した方法のことです。メタヒューリスティクスとは、同様に実験的手法ですが、個別の問題だけでなく様々な問題に応用できる汎用的な近似アルゴリズムのことです。

遺伝的アルゴリズムの処理フロー

遺伝的アルゴリズムを構成している重要な処理プロセスとして、以下の3つがあります。

- 選択（selection）
- 交叉（crossover）
- 突然変異（mutation）

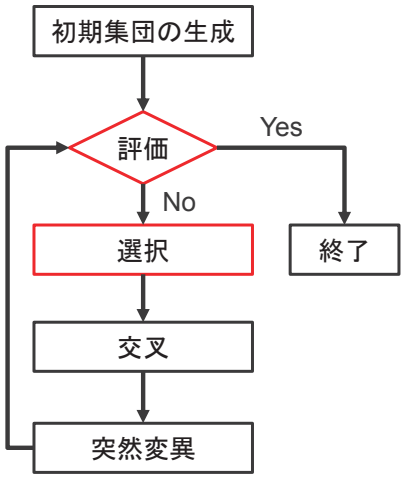


遺伝的アルゴリズムの処理フロー：選択

選択の指標となる適応度を評価します。
適応度はどれだけその個体が優れているかを示したもので、
値が大きいほど優れているとみなします。

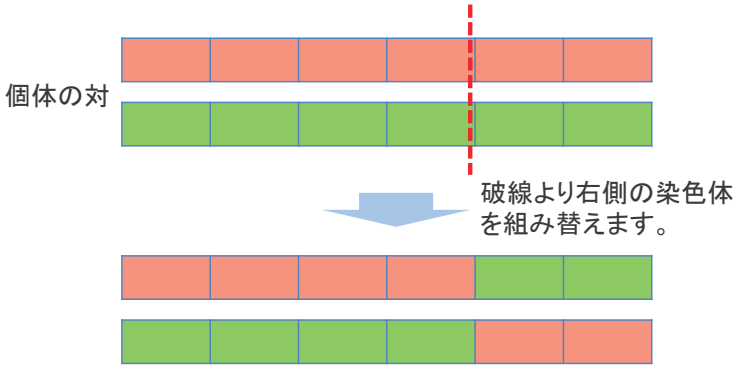
適応度を元に優秀な個体を選択します。
選択の方法として、以下のような方法があります。

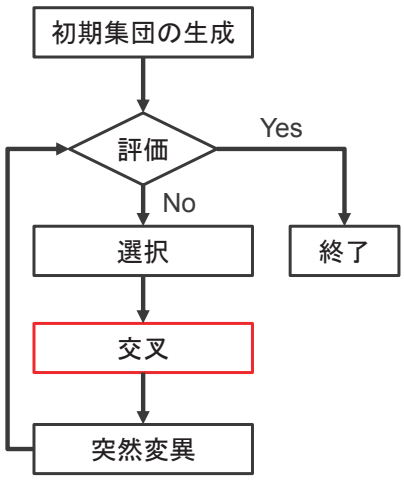
- 適応度比例戦略
  個体をランダムに選択する際に、適応度によって選ばれる確率が調整されます。
- エリート保存戦略
  最も適応度の高い固体をそのまま次世代に複製する方法です。
- トーナメント戦略
  ランダムに複数の個体を選び、最も適応度の高いものを親として残します。



遺伝的アルゴリズムの処理フロー：交叉

選択された個体間で対を作り、染色体の組み換えを行います。

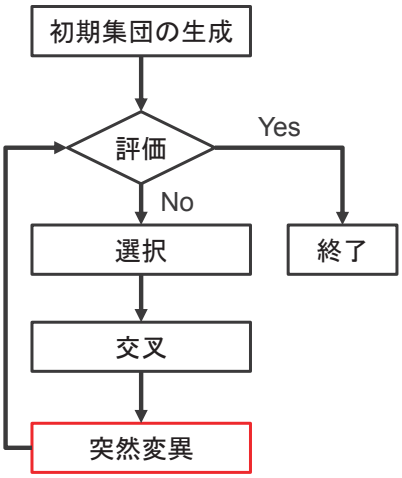




遺伝的アルゴリズムの処理フロー：突然変異

染色体に突然変異（ある確率で染色体の一部の値を変える操作）を加えます。

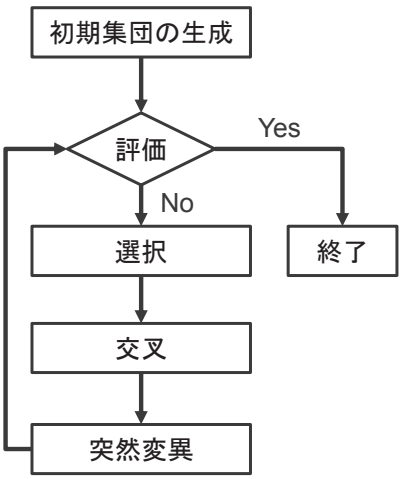
交叉だけでは個体の親に依存するような子しか生成されません。
突然変異させることで、交叉だけでは生成できない子を生成して
個体群の多様性を維持します。



遺伝的アルゴリズムの特徴

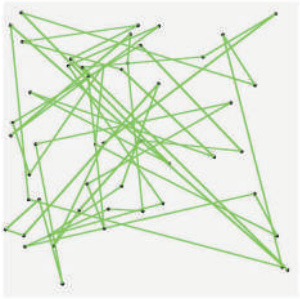
解空間構造が不明であり（決定的な優れた解法が発見されていない）、様々な可能性を実験的に全探索する必要があるような問題が存在します。
しかし、コンピュータリソースや計算時間を考えると、全探索は
適切な解放ではありません。

これまで学習してきた操作をみてわかるように、遺伝的アルゴリズムは
このような問題に対して有効です。



巡回セールスマン問題

あるセールスマンが、いくつかの地点を訪問して、スタート地点に戻ってくるとします。
各地点間の距離は分かっているとしたとき、なるべく総移動距離が短くなるように各地点を回る順番を決める、という問題が巡回セールスマン問題です。



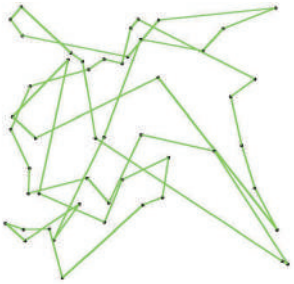


図2　第1世代から第200世代までの最短経路の変遷 (elite_rate = 0.1, mutation_rate = 0.03)　　図3　10000世代計算後の最短経路 (elite_rate = 0.1, mutation_rate = 0.03)

参照: http://blog.serverworks.co.jp/tech/2016/06/27/genetic_algorithm_traveling_salesman_ec2/

巡回セールスマン問題の計算コスト

巡回セールス問題は、地点数が増えると計算をコンピュータに実行させとしても「全探索して総移動距離が最短になる順番を見つける」という方法は現実的ではなくなります。

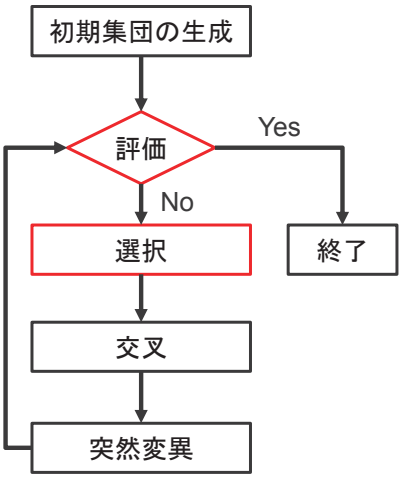
順番のパターン数は、じゅず順列と同じ考え方で計算でき、地点数がNのとき地点を訪問する順番は(N − 1)!/2 通りになります。

地点数:5	->	訪問する順番のパターン数:12
地点数:10	->	訪問する順番のパターン数:約18万
地点数:15	->	訪問する順番のパターン数:約400億

巡回セールスマン問題のような課題を解決するには、「一部を調べてみて、そこそこ移動距離が短くなる順番を見つける」という方法を取らざるを得ません。このようなときに、遺伝的アルゴリズムが応用できます。

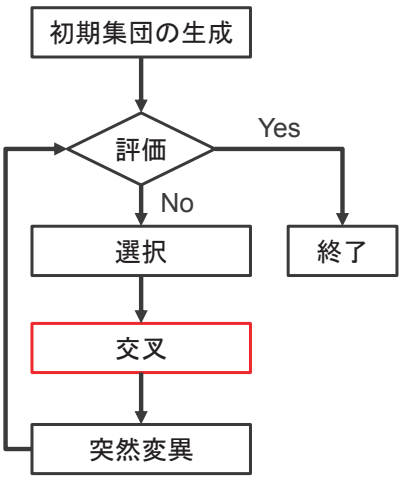
演習1：遺伝的アルゴリズム（選択）

遺伝的アルゴリズムにおける「選択」とはどのような操作を行うのか、説明してください。



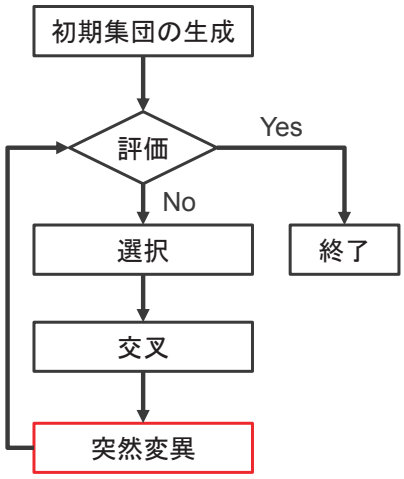
演習2：遺伝的アルゴリズム（交叉）

遺伝的アルゴリズムにおける「交叉」とはどのような操作を行うのか、説明してください。



演習3：遺伝的アルゴリズム（突然変異）

遺伝的アルゴリズムにおける「突然変異」とはどのような操作を行うのか、説明してください。



演習4：巡回セールスマン問題

巡回セールスマン問題を遺伝的アルゴリズムで解くとします。
適応度を評価するための評価関数として、どのような関数が適切でしょうか。
数式で表現してください。

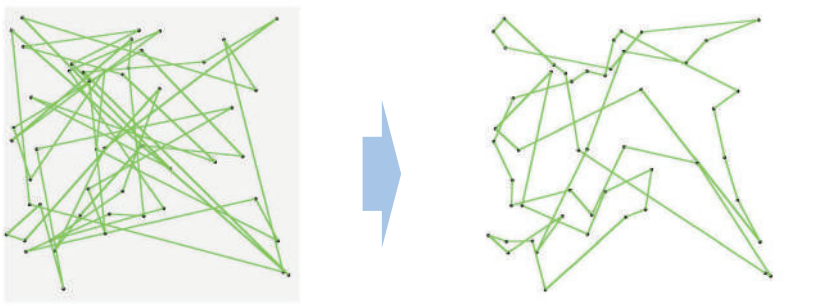


図2　第1世代から第200世代までの最短経路の変遷 (elite_rate = 0.1, mutation_rate = 0.03)　　図3　10000世代計算後の最短経路 (elite_rate = 0.1, mutation_rate = 0.03)

参照：http://blog.serverworks.co.jp/tech/2016/06/27/genetic_algorithm_traveling_salesman_ec2/

第13回：自然言語処理その1

アジェンダ

- 自然言語処理とは
- 自然言語処理の技術
  - 単語分解
  - 構文解析
  - 意味解析
  - 文脈解析

自然言語処理とは

自然言語処理とは

「自然言語」とは何か（Wikipedia：https://ja.wikipedia.org/wiki/自然言語）

- 人間がお互いにコミュニケーションを行うための自然発生的な言語である。「自然言語」に対置される語に「形式言語」「人工言語」がある。形式言語との対比では、その構文や意味が明確に揺るぎなく定められ、利用者に厳格な規則の遵守を強いる（ことが多い）形式言語に対し、話者集団の社会的文脈に沿った曖昧な規則が存在していると考えられるものが自然言語である。自然言語には、規則が曖昧であるがゆえに、話者による規則の解釈の自由度が残されており、話者が直面した状況に応じて規則の解釈を変化させることで、状況を共有する他の話者とのコミュニケーションを継続する事が可能となっている。

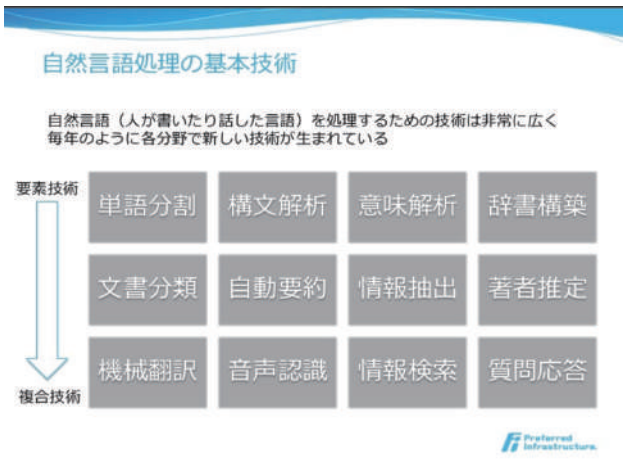
自然言語処理とは

- 「自然言語」は文化圏によって英語/日本語のように表記、文法、発音が異なります。
- 対峙している人同士、場面によって、共通認識を元にしたコミュニケーションが発生するため、全ての情報が自然言語に落ちているわけではなく、解釈が曖昧になることがあります。

- このように複雑な「自然言語」をコンピュータに処理させる技術群のことを自然言語処理といいます。

自然言語処理とは

- 自然言語処理は様々な技術で構成、整理されており、また応用分野も多岐に渡ります。



出展： https://www.slideshare.net/pfi/ss-11474303

自然言語処理の応用

- 検索、翻訳、対話など様々な製品・サービスに自然言語処理が応用されています。
- 以降のページで学習する自然言語処理の構成技術が、これらの製品・サービスでどの様に使われているのか想像してみてください。

出展（Google）

出展： https://www.apple.com/jp/siri/

自然言語処理の技術

単語分割：形態素解析

形態素解析（Wikipedia：https://ja.wikipedia.org/wiki/形態素解析）

- 形態素解析（けいたいそかいせき、Morphological Analysis）とは、文法的な情報の注記の無い自然言語のテキストデータ（文）から、対象言語の文法や、辞書と呼ばれる単語の品詞等の情報にもとづき、形態素（Morpheme, おおまかにいえば、言語で意味を持つ最小単位）の列に分割し、それぞれの形態素の品詞等を判別する作業である。

「おまちしております。」を形態素解析した結果

文字列	読み	原形	品詞の種類	活用の種類	活用形
お待ち	オマチ	お待ち	名詞-サ変接続		
し	シ	する	動詞-自立	サ変・スル	連用形
て	テ	て	助詞-接続助詞		
おり	オリ	おる	動詞-非自立	五段・ラ行	連用形
ます	マス	ます	助動詞	特殊・マス	基本形
。	。	。	記号-句点		

単語分割：N-gram解析

N-gram解析（Wikipedia：https://ja.wikipedia.org/wiki/全文検索）

- 検索対象を単語単位ではなく文字単位で分解し、後続の N−1 文字を含めた状態で出現頻度を求める方法。N の値が1なら「ユニグラム（英: uni-gram）」、2なら「バイグラム（英: bi-gram）」、3なら「トライグラム（英: tri-gram）」と呼ばれる。たとえば「全文検索技術」という文字列の場合、「全文」「文検」「検索」「索技」「技術」「術（終端）」と2文字ずつ分割して索引化を行ってやれば、検索漏れが生じず、辞書の必要も無い。

- 「全文検索技術」という文字列をバイグラムすると「全文」「文検」「検索」「索技」「技術」「術（終端）」と2文字ずつ分割されます。

単語分割：形態素解析とN-gram解析の比較

- N-gramには辞書が不要、という利点はありますが、ノイズが大きいという短所があります（例えば、「東京都」をバイグラムで分割した場合、「東京」と「京都」という結果が含まれてしまいます。）。
- もし検索エンジンにN-gramを採用すると、「東京都の天気」が知りたいのに「東京の天気」と「京都の天気」を検索してしまうかもしれません。
- 機械学習の前処理として形態素解析、N-gram解析を使用する際は、両者の特性を考慮した選択が必要です。

形態素解析とN-gramの比較

	形態素解析	N-gram
インデクシング速度	遅い	速い
インデックスサイズ	小さい	大きい
検索ノイズ	少ない	多い
検索漏れ	多い	少ない
検索速度	速い	遅い
言語依存	辞書が必要	辞書が不要

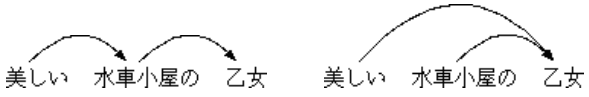
出展： https://ja.wikipedia.org/wiki/全文検索

構文解析

構文解析（Wikipedia：https://ja.wikipedia.org/wiki/構文解析）

- 構文解析（こうぶんかいせき、syntactic analysis あるいは parse）とは、文章、具体的にはマークアップなどの注記の入っていないベタの文字列を、自然言語であれば形態素に切分け、さらにその間の関連（修飾-被修飾など）といったような、統語論的（構文論的）な関係を図式化するなどして明確にする（解析する）手続きである。

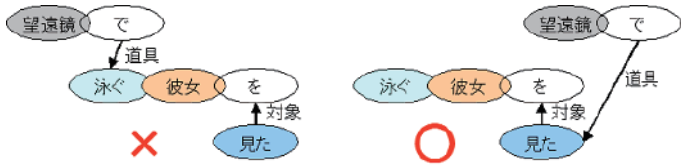
- 「美しい 水車小屋の 乙女」という文章には少なくとも2つの解釈が存在する。「水車小屋が美しい」場合と、「乙女が美しい」場合である。この場合には、意味を含めても正しい解釈がどちらであるか不明であり、その文が置かれた前後の状況、言い換えるとコンテキスト、フレーム情報などを考慮しなければ同定できない。



意味解析

- 下記の例は「望遠鏡で泳ぐ彼女を見た」という文章の解析例です。
- 人間であれば、「望遠鏡を使って泳ぐ人はいない」と判断できるため、すぐに右が正しいと分かります。
- このように、構文解析の結果が「意味」として適当かを判断することを意味解析といいます。

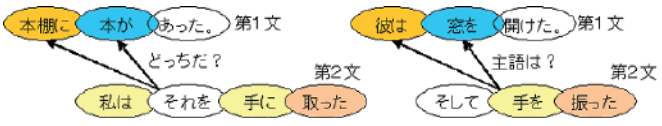
- その他の問題として、「多義性解消」というものがあります。
- 例えば「やった」という言葉には、「宿題をやった（実施した）」のような場合と、「本をやった（与えた）」という解釈の仕方があります。これをどちらか判断することも意味解析といいます。



出展： http://www.sist.ac.jp/~kanakubo/research/natural_language_processing.html

文脈解析

- 構文解析が文単位で行なわれるのに対し、複数の文にまたがる構文木作成＋意味解析を行なうのが文脈解析です。文脈解析は長い文脈に即して行なう必要があるため、単独文の意味解析よりはさらに複雑となります。
- 例えば、「それ」という代名詞が指すのは何か、という問題は文脈解析で解決します。



出展： http://www.sist.ac.jp/~kanakubo/research/natural_language_processing.html

演習

演習1：自然言語処理の応用先

- 自然言語処理が応用されている分野について調べてください。
- 現時点で技術の詳細を理解する必要はありません。自然言語処理の技術でどのようなことができるのか、漠然としたイメージで捉える程度で結構です。

演習2：自然言語処理の構成技術

- これまで単語分割、構文解析、意味解析、文脈解析の概要を学んできました。
- 皆さんが普段話す会話を分析する際、どの技術を使えば何ができるのか議論してください。

第14回：自然言語処理その2

アジェンダ

- 前回の復習
- 構文解析のアルゴリズム
  - チャート法
  - Cocke-Younger-Kasami法

前回の復習：自然言語処理の技術

単語分割：形態素解析

形態素解析（Wikipedia：https://ja.wikipedia.org/wiki/形態素解析）

- 形態素解析（けいたいそかいせき、Morphological Analysis）とは、文法的な情報の注記の無い自然言語のテキストデータ（文）から、対象言語の文法や、辞書と呼ばれる単語の品詞等の情報にもとづき、形態素（Morpheme, おおまかにいえば、言語で意味を持つ最小単位）の列に分割し、それぞれの形態素の品詞等を判別する作業である。

「おまちしております。」を形態素解析した結果

文字列	読み	原形	品詞の種類	活用の種類	活用形
お待ち	オマチ	お待ち	名詞-サ変接続		
し	シ	する	動詞-自立	サ変・スル	連用形
て	テ	て	助詞-接続助詞		
おり	オリ	おる	動詞-非自立	五段・ラ行	連用形
ます	マス	ます	助動詞	特殊・マス	基本形
。	。	。	記号-句点		

単語分割：N-gram解析

N-gram解析（Wikipedia：https://ja.wikipedia.org/wiki/全文検索）

- 検索対象を**単語単位ではなく文字単位で分解**し、後続の N-1 文字を含めた状態で出現頻度を求める方法。Nの値が1なら「ユニグラム（英: uni-gram）」、2なら「バイグラム（英: bi-gram）」、3なら「トライグラム（英: tri-gram）」と呼ばれる。たとえば「全文検索技術」という文字列の場合、「全文」「文検」「検索」「索技」「技術」「術（終端）」と2文字ずつ分割して索引化を行ってやれば、検索漏れが生じず、**辞書の必要も無い**。

- 「全文検索技術」という文字列をバイグラムすると「全文」「文検」「検索」「索技」「技術」「術（終端）」と2文字ずつ分割されます。

単語分割：形態素解析とN-gram解析の比較

- N-gramには辞書が不要、という利点はありますが、ノイズが大きいという短所があります（例えば、「東京都」をバイグラムで分割した場合、「東京」と「京都」という結果が含まれてしまいます。）。
- もし検索エンジンにN-gramを採用すると、「東京都の天気」が知りたいのに「東京の天気」と「京都の天気」を検索してしまうかもしれません。
- 機械学習の前処理として形態素解析、N-gram解析を使用する際は、両者の特性を考慮した選択が必要です。

形態素解析とN-gramの比較

	形態素解析	N-gram
インデクシング速度	遅い	速い
インデックスサイズ	小さい	大きい
検索ノイズ	少ない	多い
検索漏れ	多い	少ない
検索速度	速い	遅い
言語依存	辞書が必要	辞書が不要

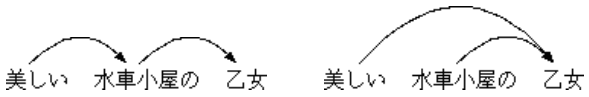
出展： https://ja.wikipedia.org/wiki/全文検索

構文解析

構文解析（Wikipedia：https://ja.wikipedia.org/wiki/構文解析）

- 構文解析（こうぶんかいせき、syntactic analysis あるいは parse）とは、文章、具体的にはマークアップなどの注記の入っていないベタの文字列を、自然言語であれば形態素に切分け、さらにその間の関連（修飾−被修飾など）といったような、**統語論的（構文論的）な関係を図式化する**などして明確にする（解析する）手続きである。

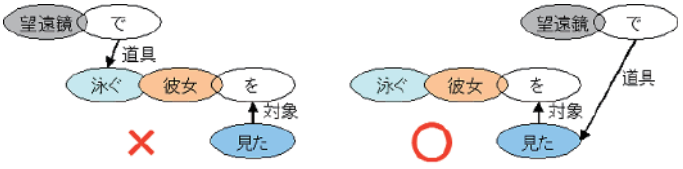
- 「美しい 水車小屋の 乙女」という文章には少なくとも2つの解釈が存在する。「水車小屋が美しい」場合と、「乙女が美しい」場合である。この場合には、意味を含めても正しい解釈がどちらであるか不明であり、その文が置かれた前後の状況、言い換えるとコンテキスト、フレーム情報などを考慮しなければ同定できない。



意味解析

- 下記の例は「望遠鏡で泳ぐ彼女を見た」という文章の解析例です。
- 人間であれば、「望遠鏡を使って泳ぐ人はいない」と判断できるため、すぐに右が正しいと分かります。
- このように、構文解析の結果が「意味」として適当かを判断することを意味解析といいます。

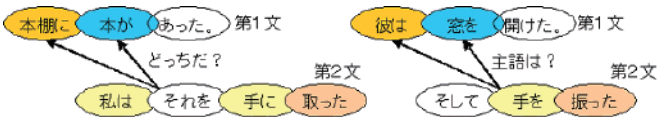
- その他の問題として、「多義性解消」というものがあります。
- 例えば「やった」という言葉には、「宿題をやった（実施した）」のような場合と、「本をやった（与えた）」という解釈の仕方があります。これをどちらか判断することも意味解析といいます。



出展： http://www.sist.ac.jp/~kanakubo/research/natural_language_processing.html

文脈解析

- 構文解析が文単位で行なわれるのに対し、複数の文にまたがる構文木作成＋意味解析を行なうのが文脈解析です。文脈解析は長い文脈に即して行なう必要があるため、単独文の意味解析よりはさらに複雑となります。
- 例えば、「それ」という代名詞が指すのは何か、という問題は文脈解析で解決します。



出展： http://www.sist.ac.jp/~kanakubo/research/natural_language_processing.html

構文解析のアルゴリズム

構文解析とは

構文解析とは、文の構文的な構造を決定することです。
句構造規則と辞書規則に則って行われます。

句構造規則は通常、次のように「合成後→合成前」という順序で表記されます。
例えば、下の表記は「AはBとCから成り立つ」という意味です。

$$A \rightarrow B\ C$$

Aが3つの構成素B、C、Dから成り立っている場合、以下のように表記できます。

$$A \rightarrow B\ C\ D$$

または

$$(A \rightarrow E\ D) \wedge (E \rightarrow B\ C)$$

※∧はandの意味

句構造規則と辞書規則

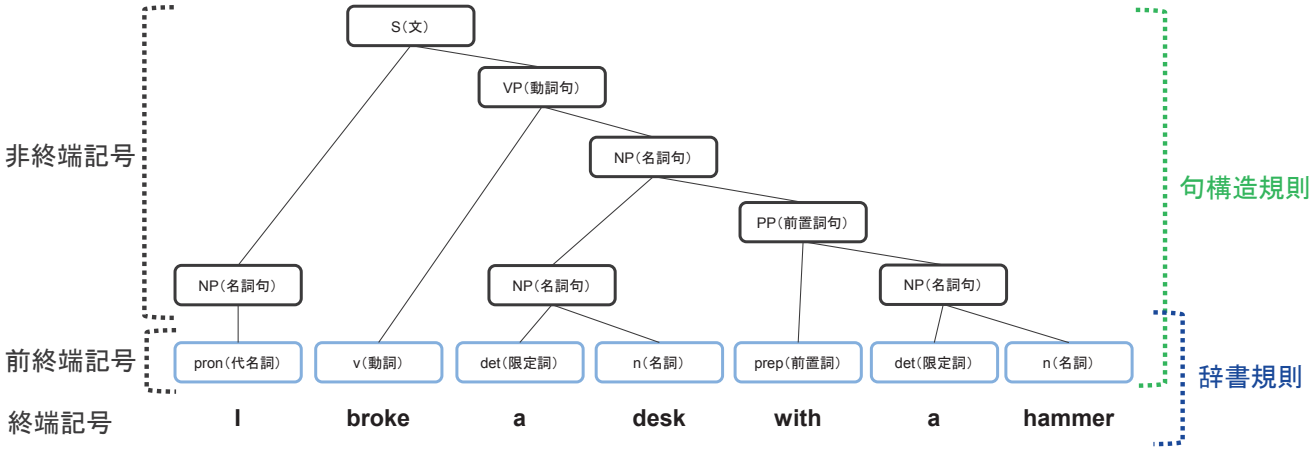
句構造規則と辞書規則には下表のような規則があります。

句構造規則	
S（文）→ NP（名詞句）VP（動詞句）	VP（動詞句）→ v（動詞）
NP（名詞句）→ pron（代名詞）	VP（動詞句）→ v（動詞）NP（名詞句）
NP（名詞句）→ det（限定詞）n（名詞）	VP（動詞句）→ VP（動詞句）PP（前置詞句）
NP（名詞句）→ NP（名詞句）PP（前置詞句）	PP（前置詞句）→ prep（前置詞）NP（名詞句）

辞書規則	
pron（代名詞）：I, You,,,	prep（前置詞）：with, above, after,,,
det（限定詞）：a, an, the,,,	n（名詞）：desk, hammer,,,
v（動詞）：broke, run, eat,,,	

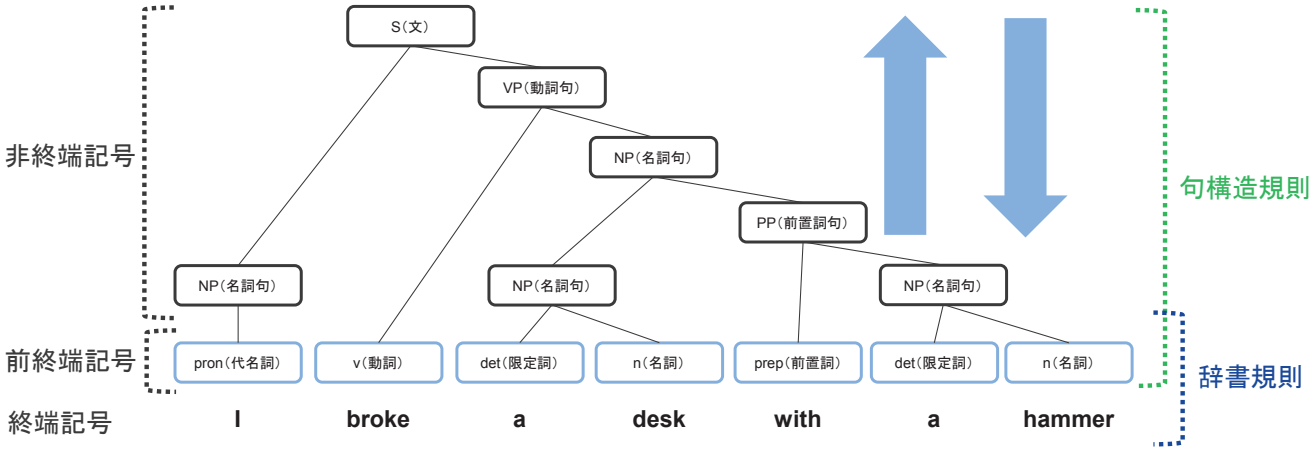
構文解析アルゴリズム

句構造規則によって非終端記号（句）と前終端記号（品詞）の構造を解析します。
辞書規則によって前終端記号（品詞）と終端記号（単語）の対応づけを行います。



構文解析アルゴリズム

S（文）から前終端記号（品詞）に向かって解析するトップダウンアルゴリズムと、前終端記号（品詞）からS（文）へと向かって解析するボトムアップアルゴリズムが存在します。



構文解析アルゴリズム

以降のページでは、構文解析アルゴリズムとして2つのアルゴリズムを学習します。

トップダウンアルゴリズム
- チャート法

ボトムアップアルゴリズム
- CYK法（Cocke-Younger-Kasami法）

チャート法

チャート法

[Wikipediaより]
チャートパーサ（英: Chart parser）は、自然言語などの曖昧な文法に向いた構文解析器の一種である。動的計画法を用い、中間的かつ仮説的な結果をチャート（chart）と呼ばれるデータ構造に格納しておき、再利用する。

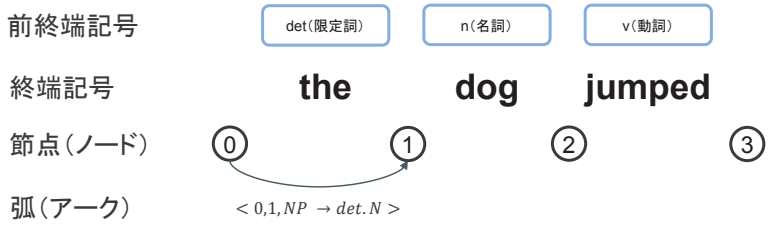
チャート法の用語

チャート法で使用する用語には、以下のものがあります。

節点（ノード）：単語間に設定する仮想的な点のことです。
弧（アーク）：節点間を結び、文の部分的な構造を表現します。

$< i, j, C \rightarrow \alpha . \beta >$

i：弧の視点、j：弧の終点、.：解析が終了している位置　を意味します。
節点iからjまで解析が終わるとαである、ということになります。
βまで解析が終わると、Cということになります。



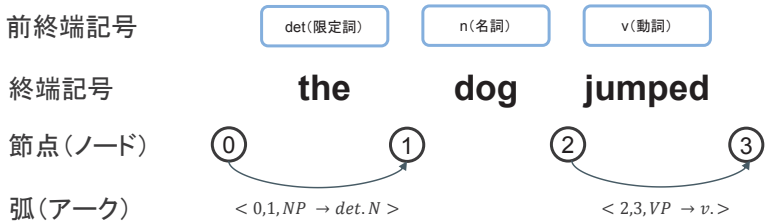
チャート法の用語

不活性弧：右辺の最後に.がある弧のことです。

例：$< 2,3,VP \rightarrow v. >$

活性弧：不活性弧以外の弧のことです。

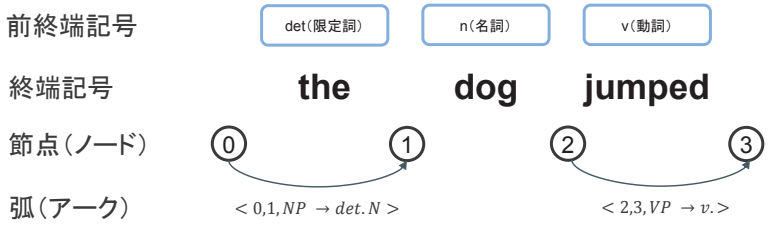
例：$< 0,1,NP \rightarrow det.N >$



チャート法の用語

チャート：ノード、弧の集合のことです。

アジェンダ：チャートに追加するべき弧のリストのことです。



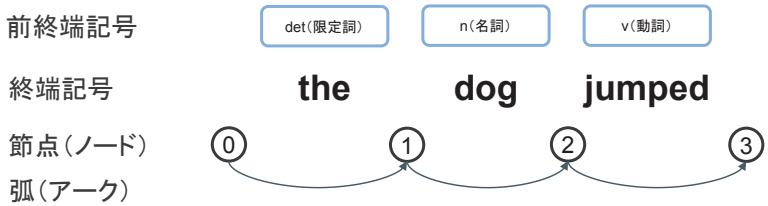
チャート法の実行（１）

入力文の各単語に対して辞書規則を適用、不活性弧を作成します。
作成した不活性弧をアジェンダに追加します。

活性弧$< 0,0,S \rightarrow .NP\ VP >$をアジェンダの先頭に追加します。

句構造規則
S -> NP VP
NP -> det n
VP -> v
VP -> v NP

辞書規則
det -> the
n -> dog
v -> jumped



アジェンダ
$< 0,0,S \rightarrow .NP\ VP >$
$< 0,1,det \rightarrow the. >$
$< 1,2,n \rightarrow dog. >$
$< 2,3,v \rightarrow jumped. >$

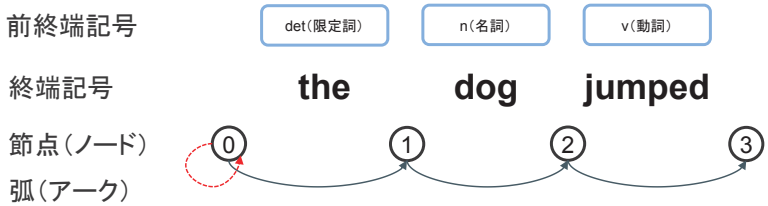
チャート法の実行（２）

アジェンダから弧を選び、チャートに追加します。

弧が活性弧$< i,j,X \rightarrow \alpha.Y\beta >$の場合、追加した弧の右側に
不活性弧$< i,j,Y \rightarrow \gamma. >$がないか探します。
不活性弧があれば結合し、なければ何もしません。

句構造規則
S -> NP VP
NP -> det n
VP -> v
VP -> v NP

辞書規則
det -> the
n -> dog
v -> jumped



$< 0,0,S \rightarrow .NP\ VP >$
不活性弧がないので、何もしません。

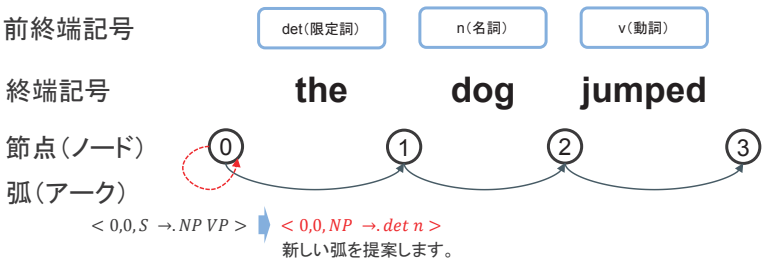
アジェンダ
$< 0,0,S \rightarrow .NP\ VP >$
$< 0,1,det \rightarrow the. >$
$< 1,2,n \rightarrow dog. >$
$< 2,3,v \rightarrow jumped. >$

チャート法の実行（３）

弧が活性弧の$< i, j, X \rightarrow \alpha . Y\beta >$場合、Yを左辺とする句構造規則がないか検索します。
該当する句構造規則があれば、新しい弧$< j, j, Y \rightarrow . \gamma >$を作ってアジェンダに追加します。

句構造規則
S -> NP VP
NP -> det n
VP -> v
VP -> v NP

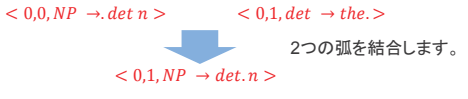
辞書規則
det -> the
n -> dog
v -> jumped



アジェンダ
~~$< 0,0,S \rightarrow . NP\ VP >$~~
$< 0,1,det \rightarrow the. >$
$< 1,2,n \rightarrow dog. >$
$< 2,3,v \rightarrow jumped. >$

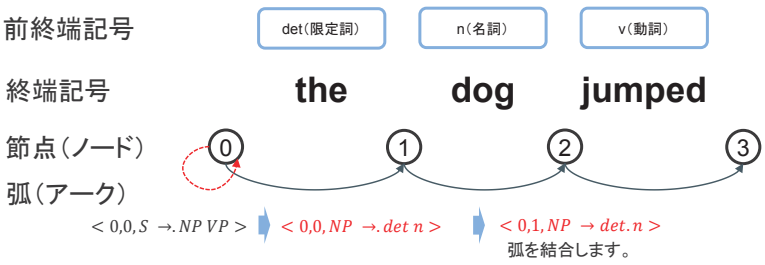
チャート法の実行（４）

新しく提案した弧が活性弧$< i, j, X \rightarrow \alpha . Y\beta >$の場合、追加した弧の右側に不活性弧$< i, j, Y \rightarrow \gamma . >$がないか探します。
不活性弧があれば結合し、なければ何もしません。



句構造規則
S -> NP VP
NP -> det n
VP -> v
VP -> v NP

辞書規則
det -> the
n -> dog
v -> jumped



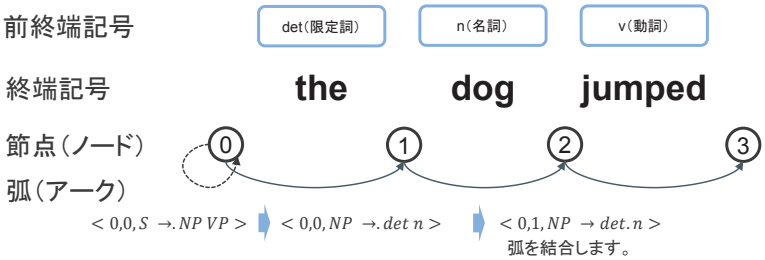
アジェンダ
~~$< 0,0,S \rightarrow . NP\ VP >$~~
$< 0,1,det \rightarrow the. >$
$< 1,2,n \rightarrow dog. >$
$< 2,3,v \rightarrow jumped. >$

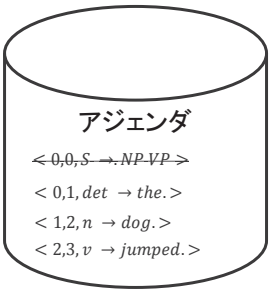
チャート法の実行（５）

アジェンダがなくなるまで(1)～(4)と同様の操作を繰り返します。
最終的に$< 0,3,S \rightarrow NP\ VP. >$となれば解析が成功したとみなします。

句構造規則
S -> NP VP
NP -> det n
VP -> v
VP -> v NP

辞書規則
det -> the
n -> dog
v -> jumped





CYK法（Cocke-Younger-Kasami法）

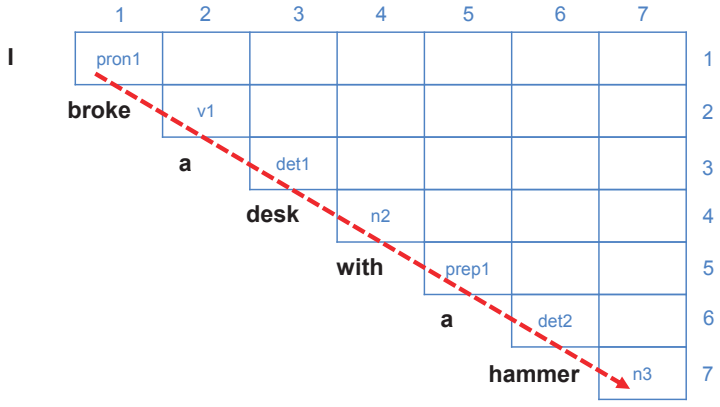
CYK法

[Wikipediaより]
CYK法（英: CYK algorithm）は、ある文字列が与えられた文脈自由文法で生成できるかを決め、生成できる場合の生成方法を求めるアルゴリズムである。CYK は Cocke–Younger–Kasami の略（それぞれ、RISCの先駆と言われる801などでも知られるジョン・コック、Daniel Younger、嵩忠雄である）。文脈自由文法の構文解析手法と捉えることもできる。このアルゴリズムは一種の動的計画法である。

CYKの実行（１）

まず、右図のように行列の対角線に
単語を配置します。

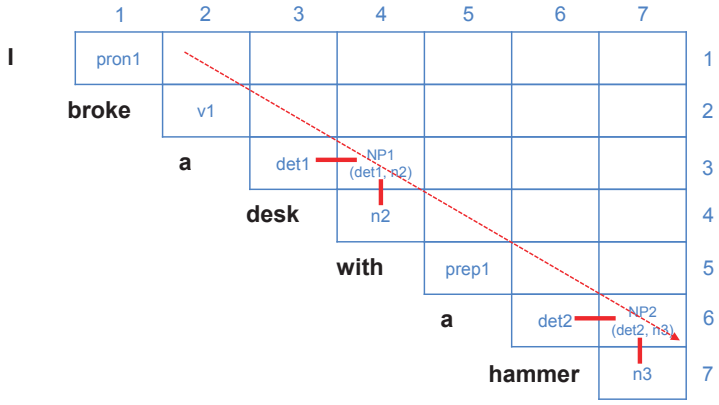
最も左の斜めの線に対して、辞書規則を
適用します。



CYKの実行（２）

次に、一つ右の対角線を見ていきます。
対角線上で句構造規則を適用できる箇所を探していきます。

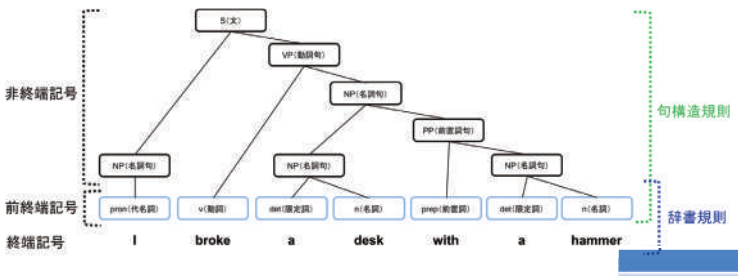
この例では、NP1とNP2が構成されます。

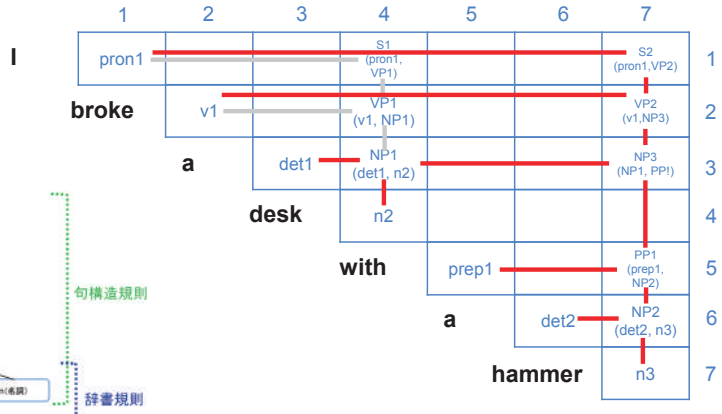


句構造規則	
S(文) → NP(名詞句) VP(動詞句)	VP(動詞句) → v(動詞)
NP(名詞句) → pron(代名詞)	VP(動詞句) → v(動詞) NP(名詞句)
NP(名詞句) → det(限定詞) n(名詞)	VP(動詞句) → VP(動詞句) PP(前置詞句)
NP(名詞句) → NP(名詞句) PP(前置詞句)	PP(前置詞句) → prep(前置詞) NP(名詞句)

CYKの実行（３）

同様の処理を繰り返し、一番右の要素に辿り着いたときに句構造が成り立っていたら解析が成功したとみなします。





句構造規則	
S(文) → NP(名詞句) VP(動詞句)	VP(動詞句) → v(動詞)
NP(名詞句) → pron(代名詞)	VP(動詞句) → v(動詞) NP(名詞句)
NP(名詞句) → det(限定詞) n(名詞)	VP(動詞句) → VP(動詞句) PP(前置詞句)
NP(名詞句) → NP(名詞句) PP(前置詞句)	PP(前置詞句) → prep(前置詞) NP(名詞句)

演習

演習1：句構造規則

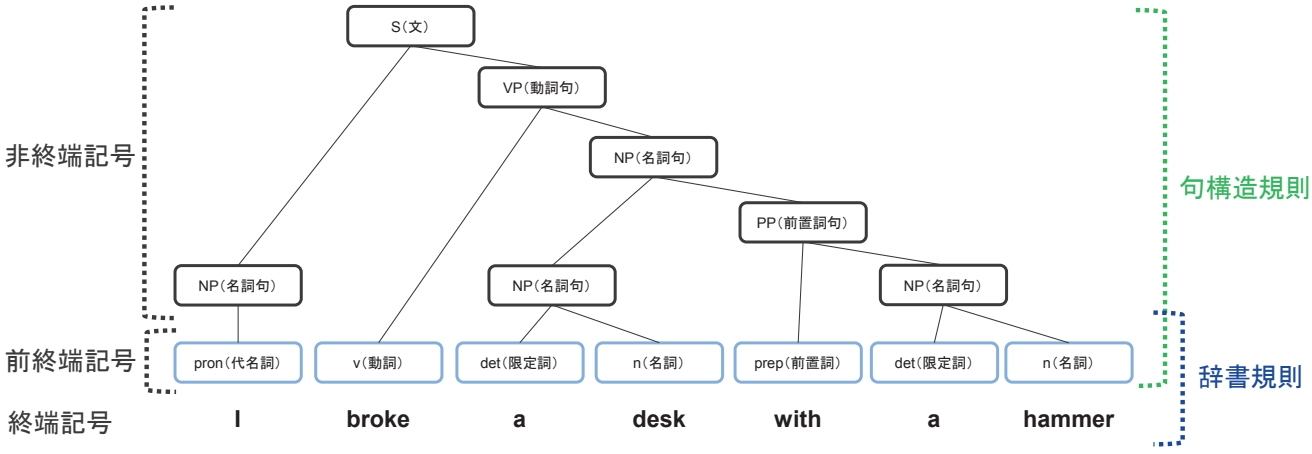
- 構文解析における句構造規則とは何か説明してください。

演習2：辞書規則

- 構文解析における辞書規則とは何か説明してください。

演習3：構文解析の全体像

- 構文解析における[句構造規則/辞書規則]と[非終端記号/前終端記号/終端記号]の関係について説明してください。



第15回：人工知能概論総復習その３

第13回：自然言語処理その1

自然言語処理とは

「自然言語」とは何か（Wikipedia：https://ja.wikipedia.org/wiki/自然言語）

- 人間がお互いにコミュニケーションを行うための自然発生的な言語である。「自然言語」に対置される語に「形式言語」「人工言語」がある。形式言語との対比では、その構文や意味が明確に揺るぎなく定められ、利用者に厳格な規則の遵守を強いる（ことが多い）形式言語に対し、話者集団の社会的文脈に沿った曖昧な規則が存在していると考えられるものが自然言語である。自然言語には、規則が曖昧であるがゆえに、話者による規則の解釈の自由度が残されており、話者が直面した状況に応じて規則の解釈を変化させることで、状況を共有する他の話者とのコミュニケーションを継続する事が可能となっている。

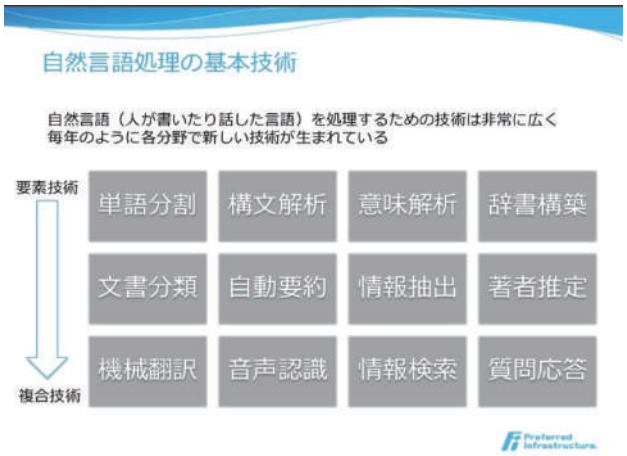
自然言語処理とは

- 「自然言語」は文化圏によって英語/日本語のように表記、文法、発音が異なります。
- 対峙している人同士、場面によって、共通認識を元にしたコミュニケーションが発生するため、全ての情報が自然言語に落ちているわけではなく、解釈が曖昧になることがあります。

- このように複雑な「自然言語」をコンピュータに処理させる技術群のことを自然言語処理といいます。

自然言語処理とは

- 自然言語処理は様々な技術で構成、整理されており、また応用分野も多岐に渡ります。



出展： https://www.slideshare.net/pfi/ss-11474303

自然言語処理の応用

- 検索、翻訳、対話など様々な製品・サービスに自然言語処理が応用されています。
- 以降のページで学習する自然言語処理の構成技術が、これらの製品・サービスでどの様に使われているのか想像してみてください。

出展（Google）

出展： https://www.apple.com/jp/siri/

単語分割：形態素解析

形態素解析（Wikipedia：https://ja.wikipedia.org/wiki/形態素解析）

- 形態素解析（けいたいそかいせき、Morphological Analysis）とは、文法的な情報の注記の無い自然言語のテキストデータ（文）から、対象言語の文法や、辞書と呼ばれる単語の品詞等の情報にもとづき、形態素（Morpheme, おおまかにいえば、言語で意味を持つ最小単位）の列に分割し、それぞれの形態素の品詞等を判別する作業である。

「おまちしております。」を形態素解析した結果

文字列	読み	原形	品詞の種類	活用の種類	活用形
お待ち	オマチ	お待ち	名詞-サ変接続		
し	シ	する	動詞-自立	サ変・スル	連用形
て	テ	て	助詞-接続助詞		
おり	オリ	おる	動詞-非自立	五段・ラ行	連用形
ます	マス	ます	助動詞	特殊・マス	基本形
。	。	。	記号-句点		

単語分割：N-gram解析

N-gram解析（Wikipedia：https://ja.wikipedia.org/wiki/全文検索）

- 検索対象を単語単位ではなく文字単位で分解し、後続の N-1 文字を含めた状態で出現頻度を求める方法。N の値が1なら「ユニグラム（英: uni-gram）」、2なら「バイグラム（英: bi-gram）」、3なら「トライグラム（英: tri-gram）」と呼ばれる。たとえば「全文検索技術」という文字列の場合、「全文」「文検」「検索」「索技」「技術」「術（終端）」と2文字ずつ分割して索引化を行ってやれば、検索漏れが生じず、辞書の必要も無い。

- 「全文検索技術」という文字列をバイグラムすると「全文」「文検」「検索」「索技」「技術」「術（終端）」と2文字ずつ分割されます。

単語分割：形態素解析とN-gram解析の比較

- N-gramには辞書が不要、という利点はありますが、ノイズが大きいという短所があります（例えば、「東京都」をバイグラムで分割した場合、「東京」と「京都」という結果が含まれてしまいます。）。
- もし検索エンジンにN-gramを採用すると、「東京都の天気」が知りたいのに「東京の天気」と「京都の天気」を検索してしまうかもしれません。
- 機械学習の前処理として形態素解析、N-gram解析を使用する際は、両者の特性を考慮した選択が必要です。

形態素解析とN-gramの比較

	形態素解析	N-gram
インデクシング速度	遅い	速い
インデックスサイズ	小さい	大きい
検索ノイズ	少ない	多い
検索漏れ	多い	少ない
検索速度	速い	遅い
言語依存	辞書が必要	辞書が不要

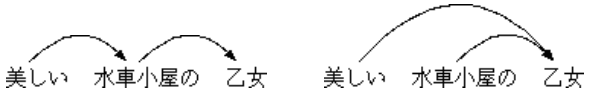
出展： https://ja.wikipedia.org/wiki/全文検索

構文解析

構文解析（Wikipedia：https://ja.wikipedia.org/wiki/構文解析）

- 構文解析（こうぶんかいせき、syntactic analysis あるいは parse）とは、文章、具体的にはマークアップなどの注記の入っていないベタの文字列を、自然言語であれば形態素に切分け、さらにその間の関連（修飾-被修飾など）といったような、統語論的（構文論的）な関係を図式化するなどして明確にする（解析する）手続きである。

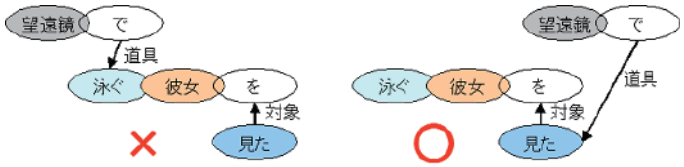
- 「美しい 水車小屋の 乙女」という文章には少なくとも2つの解釈が存在する。「水車小屋が美しい」場合と、「乙女が美しい」場合である。この場合には、意味を含めても正しい解釈がどちらであるか不明であり、その文が置かれた前後の状況、言い換えるとコンテキスト、フレーム情報などを考慮しなければ同定できない。



意味解析

- 下記の例は「望遠鏡で泳ぐ彼女を見た」という文章の解析例です。
- 人間であれば、「望遠鏡を使って泳ぐ人はいない」と判断できるため、すぐに右が正しいと分かります。
- このように、構文解析の結果が「意味」として適当かを判断することを意味解析といいます。

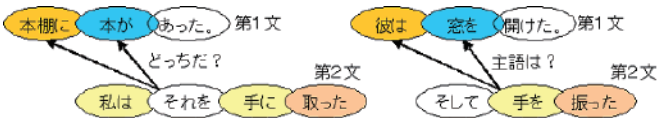
- その他の問題として、「多義性解消」というものがあります。
- 例えば「やった」という言葉には、「宿題をやった（実施した）」のような場合と、「本をやった（与えた）」という解釈の仕方があります。これをどちらか判断することも意味解析といいます。



出展： http://www.sist.ac.jp/~kanakubo/research/natural_language_processing.html

文脈解析

- 構文解析が文単位で行なわれるのに対し、複数の文にまたがる構文木作成＋意味解析を行なうのが文脈解析です。文脈解析は長い文脈に即して行なう必要があるため、単独文の意味解析よりはさらに複雑となります。
- 例えば、「それ」という代名詞が指すのは何か、という問題は文脈解析で解決します。



出展： http://www.sist.ac.jp/~kanakubo/research/natural_language_processing.html

第14回：自然言語処理その2

構文解析とは

構文解析とは、文の構文的な構造を決定することです。
句構造規則と辞書規則に則って行われます。

句構造規則は通常、次のように「合成後→合成前」という順序で表記されます。
例えば、下の表記は「AはBとCから成り立つ」という意味です。

$$A \rightarrow B\ C$$

Aが3つの構成素B、C、Dから成り立っている場合、以下のように表記できます。

$$A \rightarrow B\ C\ D$$

または

$$(A \rightarrow E\ D) \wedge (E \rightarrow B\ C)$$

※∧はandの意味

句構造規則と辞書規則

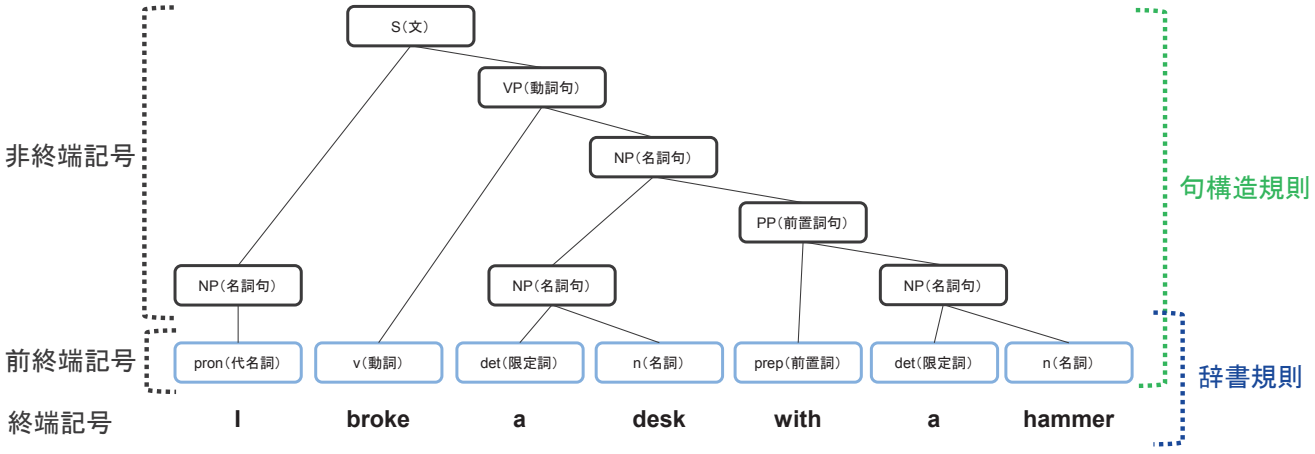
句構造規則と辞書規則には下表のような規則があります。

句構造規則	
S(文) → NP(名詞句) VP(動詞句)	VP(動詞句) → v(動詞)
NP(名詞句) → pron(代名詞)	VP(動詞句) → v(動詞) NP(名詞句)
NP(名詞句) → det(限定詞) n(名詞)	VP(動詞句) → VP(動詞句) PP(前置詞句)
NP(名詞句) → NP(名詞句) PP(前置詞句)	PP(前置詞句) → prep(前置詞) NP(名詞句)

辞書規則	
pron(代名詞):I, You,,,	prep(前置詞):with, above, after,,,
det(限定詞):a, an, the,,,	n(名詞):desk, hammer,,,
v(動詞):broke, run, eat,,,	

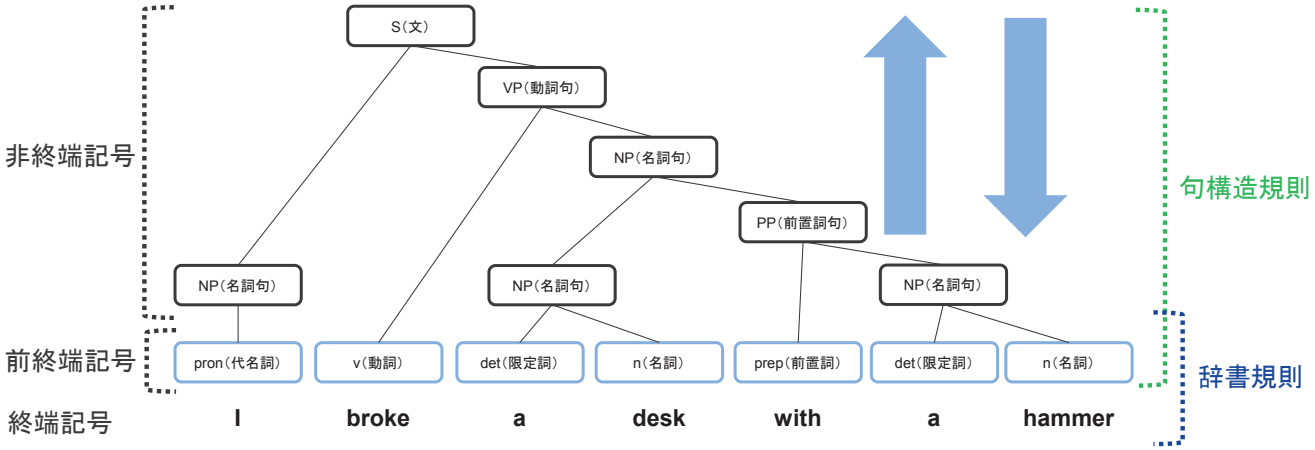
構文解析アルゴリズム

句構造規則によって非終端記号(句)と前終端記号(品詞)の構造を解析します。
辞書規則によって前終端記号(品詞)と終端記号(単語)の対応づけを行います。



構文解析アルゴリズム

S(文)から前終端記号(品詞)に向かって解析するトップダウンアルゴリズムと、前終端記号(品詞)からS(文)へと向かって解析するボトムアップアルゴリズムが存在します。



構文解析アルゴリズム

以降のページでは、構文解析アルゴリズムとして2つのアルゴリズムを学習します。

トップダウンアルゴリズム
- チャート法

ボトムアップアルゴリズム
- CYK法(Cocke-Younger-Kasami法)

チャート法

チャート法

[Wikipediaより]
チャートパーサ（英: Chart parser）は、自然言語などの曖昧な文法に向いた構文解析器の一種である。動的計画法を用い、中間的かつ仮説的な結果をチャート（chart）と呼ばれるデータ構造に格納しておき、再利用する。

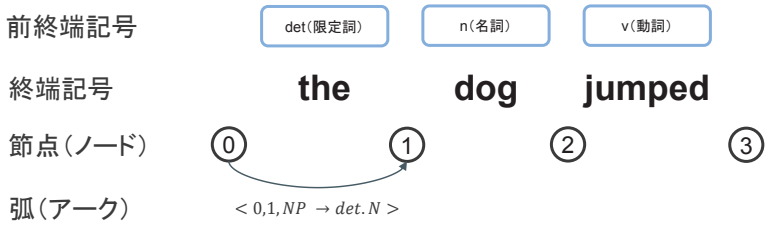
チャート法の用語

チャート法で使用する用語には、以下のものがあります。

節点（ノード）：単語間に設定する仮想的な点のことです。
弧（アーク）：節点間を結び、文の部分的な構造を表現します。

$< i, j, C \ \rightarrow \ \alpha . \beta >$

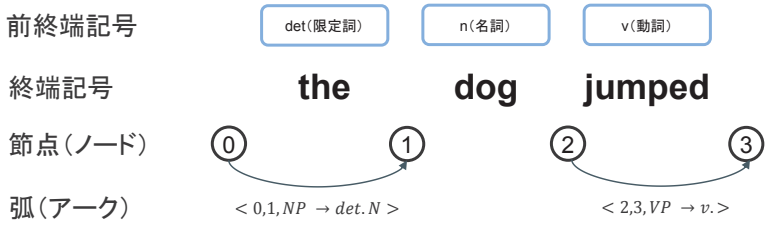
i：弧の視点、j：弧の終点、.：解析が終了している位置　を意味します。
節点iからjまで解析が終わるとαである、ということになります。
βまで解析が終わると、Cということになります。



チャート法の用語

不活性弧：右辺の最後に.がある弧のことです。
例：$< 2,3, VP \ \rightarrow v. >$

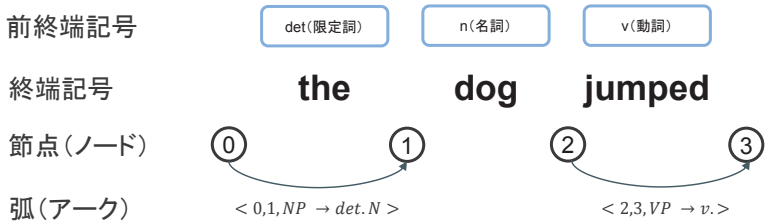
活性弧：不活性弧以外の弧のことです。
例：$< 0,1, NP \ \rightarrow det . N >$



チャート法の用語

チャート：ノード、弧の集合のことです。

アジェンダ：チャートに追加するべき弧のリストのことです。



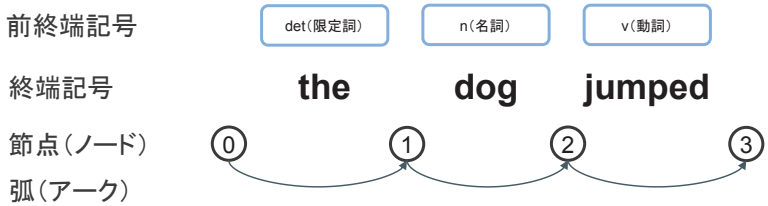
チャート法の実行（１）

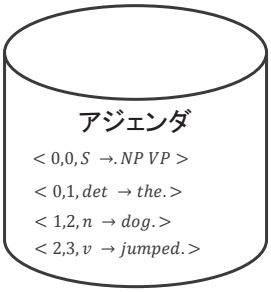
入力文の各単語に対して辞書規則を適用、不活性弧を作成します。
作成した不活性弧をアジェンダに追加します。

活性弧$< 0,0, S \rightarrow .NP\ VP >$をアジェンダの先頭に追加します。

句構造規則
S -> NP VP
NP -> det n
VP -> v
VP -> v NP

辞書規則
det -> the
n -> dog
v -> jumped





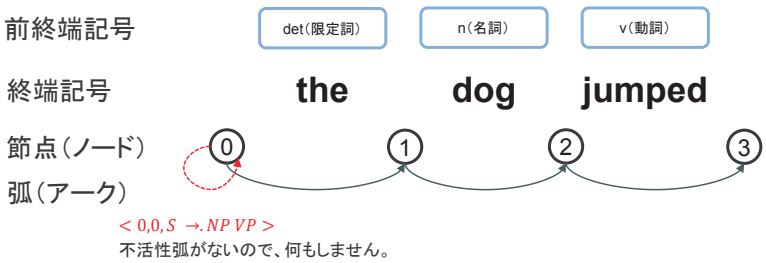
チャート法の実行（２）

アジェンダから弧を選び、チャートに追加します。

弧が活性弧$< i, j, X \rightarrow \alpha . Y\beta >$の場合、追加した弧の右側に
不活性弧$< i, j, Y \rightarrow \gamma . >$がないか探します。
不活性弧があれば結合し、なければ何もしません。

句構造規則
S -> NP VP
NP -> det n
VP -> v
VP -> v NP

辞書規則
det -> the
n -> dog
v -> jumped



$< 0,0, S \rightarrow . NP\ VP >$
不活性弧がないので、何もしません。

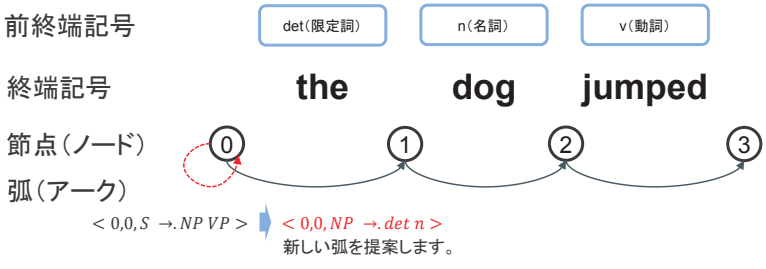
アジェンダ
$< 0,0, S \rightarrow . NP\ VP >$
$< 0,1, det \rightarrow the. >$
$< 1,2, n \rightarrow dog. >$
$< 2,3, v \rightarrow jumped. >$

チャート法の実行（３）

弧が活性弧の$< i, j, X \rightarrow \alpha . Y\beta >$場合、Yを左辺とする句構造規則がないか検索します。
該当する句構造規則があれば、新しい弧$< j, j, Y \rightarrow . \gamma >$を作ってアジェンダに追加します。

句構造規則
S -> NP VP
NP -> det n
VP -> v
VP -> v NP

辞書規則
det -> the
n -> dog
v -> jumped



$< 0,0, S \rightarrow . NP\ VP >$ ➡ $< 0,0, NP \rightarrow . det\ n >$
新しい弧を提案します。

アジェンダ
~~$< 0,0, S \rightarrow . NP\ VP >$~~
$< 0,1, det \rightarrow the. >$
$< 1,2, n \rightarrow dog. >$
$< 2,3, v \rightarrow jumped. >$

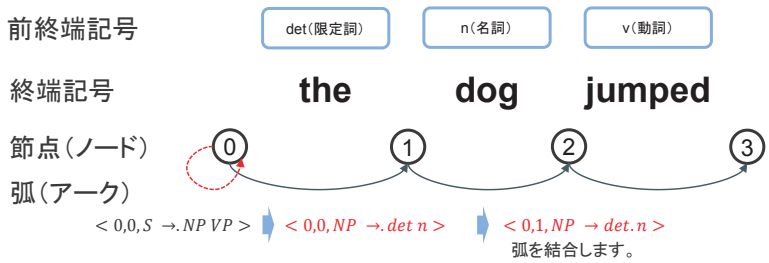
チャート法の実行（４）

新しく提案した弧が活性弧$< i,j,X \rightarrow \alpha.Y\beta >$の場合、追加した弧の右側に
不活性弧$< i,j,\mathrm{Y} \rightarrow \gamma. >$がないか探します。
不活性弧があれば結合し、なければ何もしません。

$< 0,0,NP \rightarrow .det\ n >$　　$< 0,1,det \rightarrow the. >$

2つの弧を結合します。

$< 0,1,NP \rightarrow det.n >$



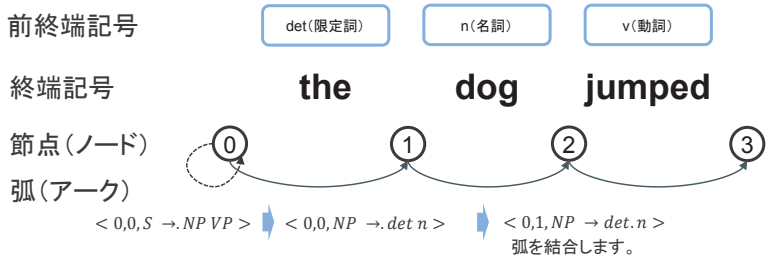
句構造規則
S -> NP VP
NP -> det n
VP -> v
VP -> v NP

辞書規則
det -> the
n -> dog
v -> jumped

アジェンダ
~~$< 0,0,S \rightarrow .NP\ VP >$~~
$< 0,1,det \rightarrow the. >$
$< 1,2,n \rightarrow dog. >$
$< 2,3,v \rightarrow jumped. >$

チャート法の実行（５）

アジェンダがなくなるまで(1)～(4)と同様の操作を繰り返します。
最終的に$< 0,3,S \rightarrow NP\ VP. >$となれば解析が成功したとみなします。



句構造規則
S -> NP VP
NP -> det n
VP -> v
VP -> v NP

辞書規則
det -> the
n -> dog
v -> jumped

アジェンダ
~~$< 0,0,S \rightarrow .NP\ VP >$~~
$< 0,1,det \rightarrow the. >$
$< 1,2,n \rightarrow dog. >$
$< 2,3,v \rightarrow jumped. >$

CYK法（Cocke-Younger-Kasami法）

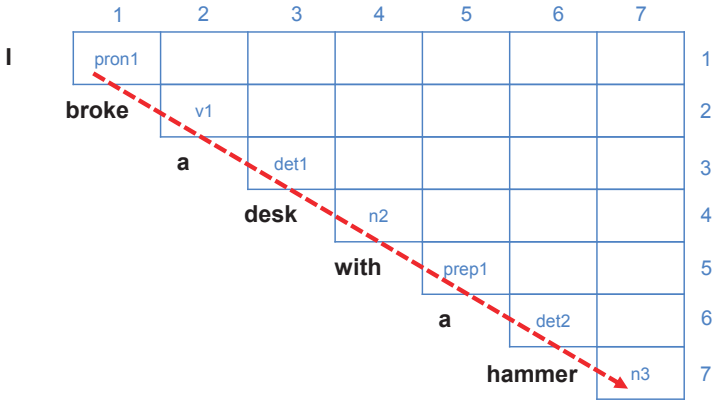
CYK法

[Wikipediaより]
CYK法（英: CYK algorithm）は、ある文字列が与えられた文脈自由文法で生成できるかを決め、生成できる場合の生成方法を求めるアルゴリズムである。CYK は Cocke–Younger–Kasami の略（それぞれ、RISCの先駆と言われる801などでも知られるジョン・コック、Daniel Younger、嵩忠雄である）。文脈自由文法の構文解析手法と捉えることもできる。このアルゴリズムは一種の動的計画法である。

CYKの実行（１）

まず、右図のように行列の対角線に
単語を配置します。

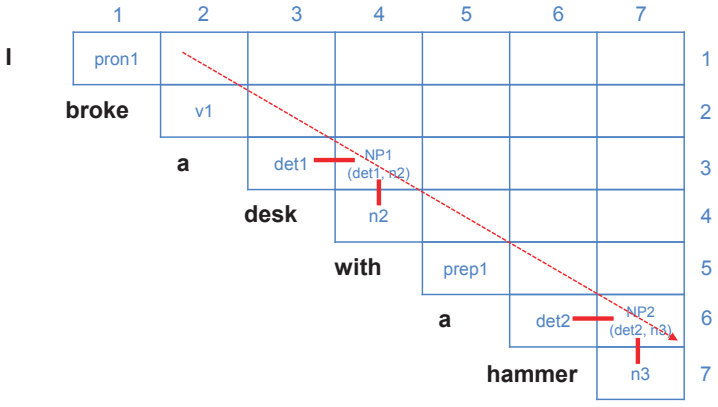
最も左の斜めの線に対して、辞書規則を
適用します。



CYKの実行（２）

次に、一つ右の対角線を見ていきます。
対角線上で句構造規則を適用できる箇所を
探していきます。

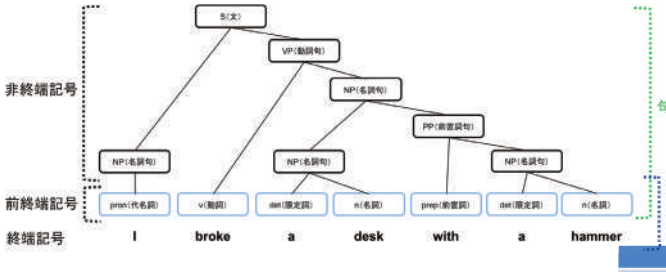
この例では、NP1とNP2が構成されます。

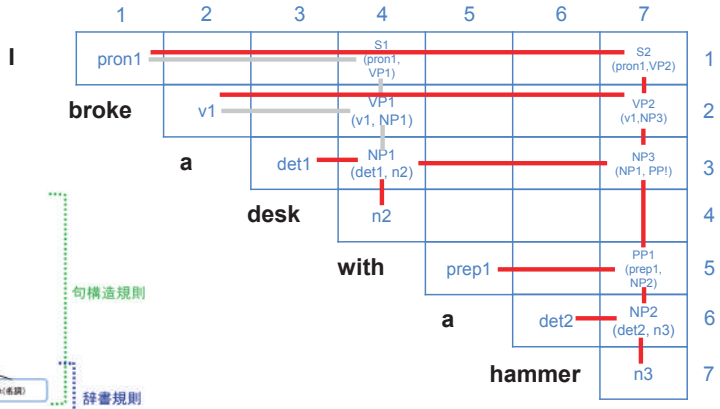


句構造規則	
S（文）→ NP（名詞句）VP（動詞句）	VP（動詞句）→ v（動詞）
NP（名詞句）→ pron（代名詞）	VP（動詞句）→ v（動詞）NP（名詞句）
NP（名詞句）→ det（限定詞）n（名詞）	VP（動詞句）→ VP（動詞句）PP（前置詞句）
NP（名詞句）→ NP（名詞句）PP（前置詞句）	PP（前置詞句）→ prep（前置詞）NP（名詞句）

CYKの実行（３）

同様の処理を繰り返し、一番右の要素に
辿り着いたときに句構造が成り立っていたら
解析が成功したとみなします。





句構造規則	
S(文) → NP(名詞句) VP(動詞句)	VP(動詞句) → v(動詞)
NP(名詞句) → pron(代名詞)	VP(動詞句) → v(動詞) NP(名詞句)
NP(名詞句) → det(限定詞) n(名詞)	VP(動詞句) → VP(動詞句) PP(前置詞句)
NP(名詞句) → NP(名詞句) PP(前置詞句)	PP(前置詞句) → prep(前置詞) NP(名詞句)

演習

演習1：句構造規則

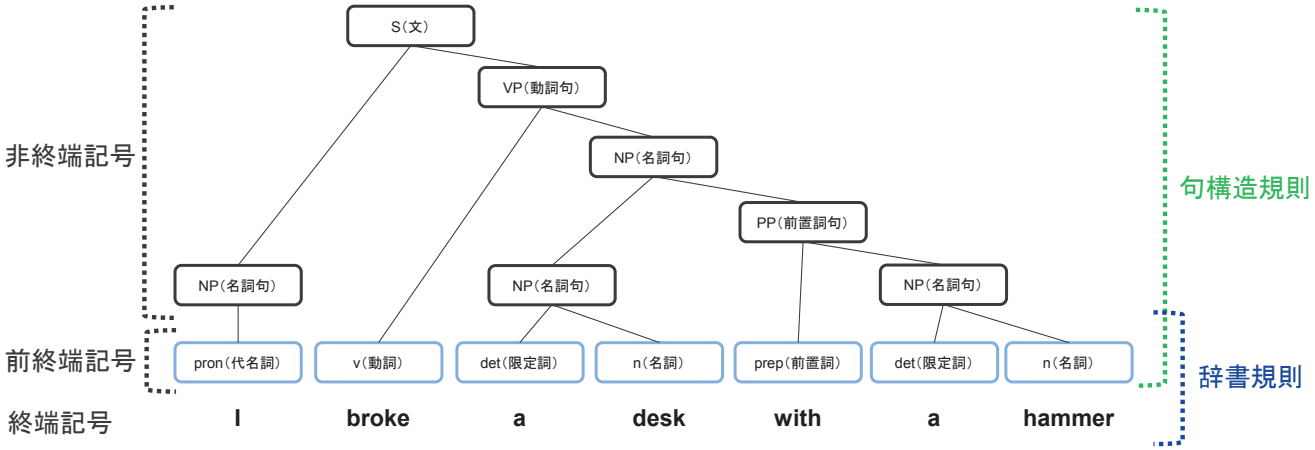
- 構文解析における句構造規則とは何か説明してください。

演習2：辞書規則

- 構文解析における辞書規則とは何か説明してください。

演習3：構文解析の全体像

- 構文解析における[句構造規則/辞書規則]と[非終端記号/前終端記号/終端記号]の関係について説明してください。



令和２年度「専修学校による地域産業中核的人材養成事業」
Society5.0 実現のための IT 技術者養成モデルカリキュラム開発と実証事業

令和２年度「専修学校による地域産業中核的人材養成事業」
Society5.0 実現のための IT 技術者養成モデルカリキュラム開発と実証事業

人工知能概論

令和３年２月

学校法人電子学園（日本電子専門学校）
〒169-8522　東京都新宿区百人町 1-25-4
TEL 03-3369-9333　FAX 03-3363-7685