平成30年度文部科学省委託
「専修学校による地域産業中核的人材養成事業」

機械学習Ⅰ

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

平成30年度文部科学省委託
「専修学校による地域産業中核的人材養成事業」

機械学習 I

目 次

第1回：データ活用概論

データ活用全般に関しての説明

アジェンダ

- 講義で取り扱う内容
- データ分析とは？
- データ分析に関わる重要キーワードの解説
  - キーワード１：ビッグデータ
  - キーワード２：統計
  - キーワード３：データサイエンティスト
  - キーワード４：人工知能
  - キーワード５：機械学習
  - キーワード６：深層学習
- 演習課題

全15回の講義について

- 主にデータ分析の手法を中心に講義を実施する
  - プログラミング言語としてはPython
  - Pythonの各種ライブラリを利用してデータ分析に必要なスキルの習得を目指す
    - 基本的な統計解析
    - データの可視化
    - 機械学習
    - etc...
  - 第2回以降の講義で詳細を取り扱う

データ分析とは？

分析とは？

Wikipediaの「分析」によると…

1. ある物事を分解して、それらを成立している成分・要素・側面を明らかにすること
2. 物質の鑑識・検出、また化学的組成を定性的・定量的に鑑別すること。
3. 概念の内容を構成する諸徴表を各個別に分けて明らかにすること。
4. 証明するべき命題から、それを成立させる条件へ次々に遡っていくやり方。

つまり、データ分析とは・・・

データを用いてその背後にある

（かもしれない）

事象を明らかにしていくこと

データ分析が必要とされる背景

- あらゆる物事のデジタル化
  - 「データ」を集めることが容易となる

  ⇒ ビッグデータ

- 「勘、経験、度胸」による従来型の意思決定プロセスからの脱却
  - 属人的で再現性が低い
  - データ分析による意思決定は客観性が高く、再現可能性もある

  ⇒ 統計

データ分析のその先に

- データ分析の実行者は新しい職業
  - 必要とされるが市場にはまだあまり存在しない

  ⇒ データサイエンティスト

- データによる事象の理解/分析のその先へ
  - データとアルゴリズムを用いた知的活動の代替

  ⇒ 人工知能
  機械学習
  深層学習

データ分析に関わる重要キーワードの解説

キーワード１：ビッグデータ

Wikipediaの「ビッグデータ」によると…

1. ある物事を分解して、それらを成立している成分・要素・側面を明らかにすること
2. 物質の鑑識・検出、また化学的組成を定性的・定量的に鑑別すること。
3. 概念の内容を構成する諸徴表を各個別に分けて明らかにすること。
4. 証明するべき命題から、それを成立させる条件へ次々に遡っていくやり方。

つまり、ビッグデータとは・・・

文字通りものすごく大量で

種類も様々なデータ

ビッグデータが注目される背景

1. デジタル化によるデータ収集の容易さ
   - ➢ 色々なデータを集めてみよう

1. クラウドによるデータ保管コストの低さ
   - ➢ たくさんのデータをとりあえずは残しておこう

1. 統計的な分析による有用性
   - ➢ データ分析はビジネスに活用できる

ビッグデータが注目される背景

1. デジタル化によるデータ収集の容易さ
   - ➢ 色々なデータを集めてみよう

1. クラウドによるデータ保管コストの低さ
   - ➢ たくさんのデータをとりあえずは残しておこう

1. 統計的な分析による有用性
   - ➢ データ分析はビジネスに活用できる

データ収集の容易さ

1. デジタル化されたサービスはログとして大量のデータ（主に行動ログ）が簡単に取得される
2. 会員情報や購買履歴などのトランザクション情報もデータとして取得される

- 2つの組み合わせることでデジタル上における１人１人の行動がつぶさに記録される

ビッグデータが注目される背景

1. デジタル化によるデータ収集の容易さ
   - ➢ 色々なデータを集めてみよう

1. クラウドによるデータ保管コストの低さ
   - ➢ たくさんのデータをとりあえずは残しておこう

1. 統計的な分析による有用性
   - ➢ データ分析はビジネスに活用できる

データ保管コストの低さ

- クラウドはデータ保管コストが非常に低い
  - 例えばAWSだと最安値は1TB/月までは$0.0114/GB
    - 1TB保管しても月1000円程度！（※データ転送などは別途費用が発生）
    - バックアップなどの対応もされているし、今後もさらに費用は下がるはず

- クラウド以前は例えば「〇〇ヶ月分だけログを残しましょう」と言ったルールを決めてデータの保管は運用されていた
- クラウドの登場により、ログはとりあえずクラウドのストレージに保存しておこう
  - 必要かどうかは後で判断すれば良い

ビッグデータが注目される背景

1. デジタル化によるデータ収集の容易さ
   - ➢ 色々なデータを集めてみよう

1. クラウドによるデータ保管コストの低さ
   - ➢ たくさんのデータをとりあえずは残しておこう

1. 統計的な分析による有用性
   - ➢ データ分析はビジネスに活用できる

データ分析の有用性

- 2010年ごろまではそもそも大量のデータを保有できていたのが一部の大規模Webサービスの運営企業だけだった
  - ○ ソーシャルゲームのヒット、アドテクノロジーの対等により、自社で大規模なWebサービスを運営していなくても大量のデータが手に入るようになった

- ソーシャルゲームの高収益を支える要因の一つに統計的なデータ分析があることが広まり、一気にデジタル領域でのデータ分析に関する認識が広がった

これまでのビッグデータのトレンド

- 前項の通り日本ではソーシャルゲーム、アドテクノロジーを中心に広がった
  - どちらかというとWeb系のライトな領域で広がった
- 海外では、同時に金融や物流、小売などのWeb系にとどまらず、ビジネス全体にデータ活用が広まっていた
- 日本でも近年ようやく、非Web系の企業にもデータ活用のトレンドが広まりつつある印象
  - いくつかの先進的な企業が取り組みを行い、成功事例が外に出始めている印象

キーワード２：統計

Wikipediaの「統計」によると...

- 統計（とうけい、statistic）は、現象を調査することによって数量で把握すること、または、調査によって得られた数量データ(統計量)のことである。

つまり、統計とは・・・

現象をデータから読み解くための

技術やその結果そのもの

統計学はその技術を

体系立てた学問

なぜ統計が必要か？

- データは簡単に、かつたくさん手に入るようになった

- 次に必要なのは、データから「何が起きているのか」を正確に理解し、「次の行動の意思決定材料」とする方法
- 手元にあるデータを正しく理解するために道具としての統計が必要となってくる

統計学について

Wikipediaの「統計学」によると...

- 統計学は、経験的に得られたバラツキのあるデータから、応用数学の手法を用いて数値上の性質や規則性あるいは不規則性を見いだす。統計的手法は、実験計画、データの要約や解釈を行う上での根拠を提供する学問であり、幅広い分野で応用されている。 現在では、医学（疫学、EBM）、薬学、経済学、社会学、心理学、言語学など、自然科学・社会科学・人文科学の実証分析を伴う分野について、必須の学問となっている。また、統計学は哲学の一分科である科学哲学においても重要なひとつのトピックスになっている。

つまり、統計学とは・・・

データを正しく解釈し、意思決定を

行うために必須の学問

これまでは学問領域での適用に

とどまってきたが、ビジネス活動にも

必要とされるようになり始めている

キーワード３：データサイエンティスト

- 21世紀で最もセクシーな職業の一つと言われている
  - 出典：ハーバードビジネスレビュー

- 日本では、「データサイエンティスト」という用語だけが先行してしまい、その職業イメージがまだ定まりきっていない状態
  - 一般社団法人データサイエンティスト協会などが立ち上がり、業界としてしっかりと職業を定義し、守っていこうという流れがある

データサイエンティストの仕事

- 業務として必要なスキルは、以下3つに集約されてくる

1. ビジネス上の課題を整理し、分析の文脈に落とし込み、分析結果をビジネス上の言葉に置き換えるコンサルティング力
2. 統計や機械学習を駆使した分析力
3. データやアルゴリズムを取り扱うためのエンジニアリング力

３つのスキルをすべて持つことは可能

- 方向性の異なるスキルのため、現実的には同時に全てのスキルを要することは非常に難しいと言える
  - 実際には1つ+αのスキルを持つ人がデータサイエンティストを名乗っていることが多い

- そのため、実務ではチームワークが大切
  - チームとして3つのスキルを持てるようにするうことがより重要

キーワード４：人工知能

- 自動運転車や囲碁のプロに勝利するなど、注目度が非常に高い
- 人工知能そのものは立場によって様々な定義がある

- 最近普及している「人工知能」は以下の文脈であることが多い
  - 「人間の特権だと思われていた知的活動」を代替するテクノロジー
- 人間の活動を代替できるため、ビジネス適用の可能性も探ることができ、非常に有望
  - 画像、自然言語、音声などを対象にしたアプリケーションが登場してきている

キーワード５：機械学習

- 与えられたデータからその背後にあるルールを見つけ出すアルゴリズム
- アルゴリズムの分類としては下記のようなものがある
  - 教師あり学習
  - 教師なし学習
  - 強化学習

- あらゆるデータやタスクに対して万能な機械学習のアルゴリズムは存在しない

キーワード６：深層学習

- 最近最も注目を浴びている機械学習の一種
  - ニューラルネットワークの派生系と言える

- 画像処理などの一部のタスクで従来の成果を大きく超える成果を上げている
  - タスクによっては人間以上の精度とも言える

- 但し、膨大な学習データと計算リソースが必要となる

演習課題

1. データを活用している取り組みについて調べて見ましょう。（全ての情報が公開されているわけではないため、ある程度想像もしながら）
   - ➢ どんなデータを使っていて
   - ➢ どんな分析を行っていて
   - ➢ 分析結果をどう活用しているか

1. １で調べたことをもとにして、あなたがデータを活用する取り組みを行うとしたら、どんなことを行うか自由に考えて回答してみてください。

演習課題（回答例）

1. データを活用している取り組みについて調べて見ましょう。（全ての情報が公開されているわけではないため、ある程度想像もしながら）
   - ➢ ソーシャルゲーム業界では、プレイヤーの行動ログを保存している
   - ➢ 例えば、ユーザーが離脱しているポイントなどをデータから分析
   - ➢ 分析結果から、難易度の調整であったり、ゲームシステムの変更を行っている

1. １で調べたことをもとにして、あなたがデータを活用する取り組みを行うとしたら、どんなことを行うか自由に考えて回答してみてください。
   - ➢ ここはご自身のビジネスに照らすなどして自由に考えて見てください。

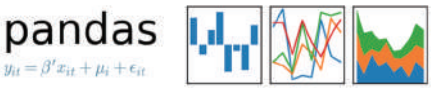
第2回： データ分析ソフトウェアの紹介と分析環境の構築

データ分析に関するソフトウェアを紹介し
講義で利用する分析環境を構築する

アジェンダ

- 分析環境の概要
- 分析環境のプログラミング言語やライブラリ紹介
  - Python
  - Jupyter
  - Pandas
  - Scipy
  - Statsmodels
  - NumPy
  - matplotlib
  - scikit-learn

本講義で使う分析環境



Python

- 高水準のプログラミング言語の一種
  - 動的な型付け
  - ガベージコレクション
  - マルチパラダイム（手続き型、オブジェクト指向、関数型、etc）
- オランダ人のGuido Van Rossumによって開発
- データ分析、機械学習分野では非常によく使われている言語
  - Webアプリケーション、デスクトップアプリケーションも開発が可能
- インタラクティブシェルが付属されているため、トライアンドエラーで作業が進めやすい

Jupyter

- Webブラウザから Pythonなどのプログラミング言語を実行させる環境
- もともとは、Pythonのインタラクティブシェルをより強力にしたIPythonが始まり
  - セル単位でのプログラム実行とタブ補完
  - オブジェクトの調査とシェルコマンドの実行
  - etc...
- IPythonをWeb経由に実行させられるものとして、IPython Notebookが登場
  - テーブルやグラフ、数式の埋め込み
  - Markdownの埋め込み
  - .ipynb形式により、分析過程自体を再実行性を担保して保存と共有が可能に
- IPython NotebookをPython以外の言語にも対応させたものがJupyter Notebook
  - Python
  - R
  - Julia

NumPy

- NumPyはPythonのライブラリの一つ
- 大規模な多次元配列や行列の操作をサポート
- 特に行列演算が必要となる領域（画像処理）などで重宝されている
  - 機械学習も深層学習も行列計算が必要になるため、必須ライブラリの一つ
- Numpy自体を使う機会はそこまで多くないかもしれないが、分析系ライブラリの背後ではNumpyが使われていることが多い

Pandas

- PandasはPythonのライブラリの一つ
- データ分析を支援するための機能を多数提供している
- DataFrameと呼ばれるテーブル形式のデータフォーマットを提供
  - SQLライクなデータ操作を宣言的に記述できる
  - DataFrameと外部データソース間で相互に読み書き可能
- 時系列データもサポート

matplotlib

- matplotlibはPythonのライブラリの一つ
- グラフの描画機能を提供している
  - 描画したグラフを多数の画像形式に変換可能
- Pythonにも様々なグラフ描画ライブラリはあるが最も普及しているものの一つ

SciPy

- SciPyはPythonのライブラリの一つ
- 科学計算のための機能を多数提供している
  - 統計
  - 最適化
  - 線形代数
  - フーリエ変換
  - 信号・画像処理
  - 遺伝的アルゴリズム
  - 微分方程式
  - etc..

scikit-learn

- scikit-learnはPythonのライブラリの一つ
- 機械学習に関しての種々の機能を提供している
  - クラスタリング
  - ロジスティック回帰
  - サポートベクターマシン
  - ランダムフォレスト
  - etc...

分析環境の構築

- 前提としてはPythonは3系の最新版である3.6系
- 構築方法としては下記2つ
  - 素のPythonに必要となるライブラリをインストールしていく方法
    - Pythonの環境を自力で構築できる方向け
  - Anaconda(miniconda)を利用する方法
    - Pythonの環境を自力で構築する自身がない方向け
    - インストーラを使って必要なソフトウェア一式のインストーラを行う

1. 素のPythonからインストール

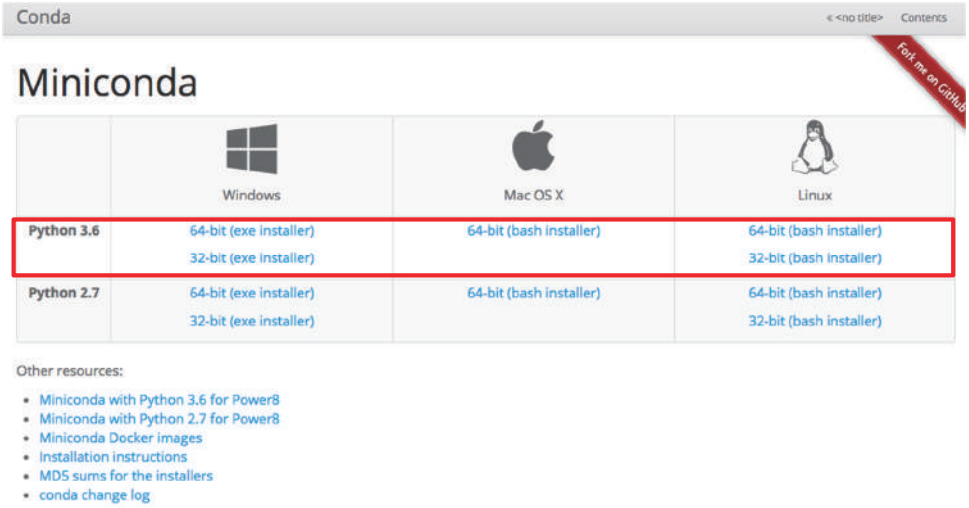
- Pythonにはパッケージ管理システムであるpipが標準で付属しているため、pipを利用して各種ライブラリをインストールする
  - Python3系そのもののインストールは割愛

```
$ pip install jupyter numpy pandas scipy matplotlib scikit-learn
```

2. Anaconda(miniconda)を利用してインストール

- AnacondaはPython及び分析に必要なライブラリ一式を提供してくれるパッケージ
- minicondaはAnacondaの中で分析に必要な最小限のみ集めた軽量のディストリビューション
  - Anacondaのダウンロードページ（https://conda.io/miniconda.html）から使用しているOSに合わせたインストーラを取得してダウンロードする
  - 今回はPython3.6系が前提のため、Python3.6系のインストーラを利用する

2. Anacondaを利用してインストール（続き）



3. minicondaの使い方

- 環境を作る

```
$ conda create -n analytics
```

- 環境のアクティベート

```
$ activate analytics
```

- 環境内でのパッケージインストール

```
$ pip(conda) install jupyter numpy pandas scipy matplotlib scikit-learn
```

- 環境の終了

```
$ deactivate analytics
```

以後の作業

- Jupyter notebook上での作業を推奨とします。
- コマンドを実行したディレクトリをルートとしてjupyterが起動します。

```
$ jupyter notebook
```

演習課題

- 実際に分析環境を構築すること
- 環境を構築できたら、jupyter notebookが動作することを確認すること
- 各ライブラリのホームページに行くと、チュートリアルが用意されている。適当に試してみることで正しくインストールが出来ていることや、Pythonのプログラムに慣れること

第3回： Python基礎

Pythonの基礎

アジェンダ

- Pythonの基礎文法
  - 四則演算
  - 文字列
  - 変数
  - リスト
  - 制御構文
    - if文による条件分岐処理
    - for文によるループ処理
  - 関数
- 演習課題

四則演算

- 電卓の用に基本的な演算を実行可能
  - 足し算
  - 引き算
  - 掛け算
  - 割り算
- 累乗の計算などもできるため、通常の電卓よりもリッチな計算ができる

```
2 + 2
>>> 4

2 * 5
>>> 10
```

文字列

- シングルクォート、もしくはダブルクォートで囲うことで文字列にできる
- 日本語も扱える

```
“hoge”
>>> “hoge”

‘日本語もOK’
>>> 日本語もOK
```

print関数

- 変数の内容や計算結果を表示するための便利な関数

```
print(2+2)
>>> 4

print(‘hoge’)
>>> hoge
```

変数

- 様々な値が代入可能な箱のようなもの
- CやJavaのように事前に変数の型を指定する必要はない
- 変数の内容はprint関数で表示することができる
- 変数は再代入も可能

```
a = 1
b = ‘hoge’

a = ‘fuga’
```

文字列再び

- 文字列に対する演算
  - ‘+’演算で文字列の連結
  - ‘*’演算で繰り返し
- インデクスでのアクセス
  - 文字単位でのアクセスが可能
  - インデクスは0番から始まる
  - インデクスに負の値を入れると文字列の終端からのアクセスとなる
  - 範囲指定で部分文字列を取り出せる
- immutableな変数
  - 部分文字列の変更は不可能
  - インデクスでアクセスし、再代入しようとするとエラーとなる

リスト

- 複数の値をまとめて保持することのできるデータ構造
  - CやJavaの配列のようなデータ構造
  - 数値でも文字列でも保持可能で、1つのリストに数値と文字列を混ぜて保持することも可能
- 文字列と同じようにインデクスによるアクセスが可能
  - 文字列と異なる点はmutableであるため、再代入が可能である点

```
nums = [1, 2, 3, 4, 5]

words = [‘hoge’, ‘fuga’]
```

制御構文

- プログラムのバリエーションを増やす上で必須の構文
  - if文による条件分岐
  - for文によるループ

if文による条件分岐

- ifの後に条件式を書く
  - 評価した結果、条件に該当する場合は処理が動く
    - CやJavaと同じ
  - 一方、CやJavaとは異なり、インデントで揃えたコードブロック単位で条件節が決まる
- ifに該当しない場合、必ず通る条件として、elseを使うことができる
  - CやJavaと同じ
- elifを利用して、複数の条件判定を行うことも可能

```
if x == 50:
        print(‘50です’)
else:
        print(‘50ではありません’)
```

for文による繰り返し

- Pythonのfor文はリストを走査してループさせるのが基本
  - リストの要素を取り出しながら処理を進めていく
- CやJavaのforループのような処理はできないのか？
  - ループの変数を任意の数までインクリメントしながらループ処理の実行
  - range関数を使えば実現可能
- 多重ループ
  - ループ内でループをまわすことができる

```
nums = [1, 2, 3, 4, 5]
for num in nums:
        print(num)
```

break文とcontinue文

- break文
  - その時点でループを打ち切る
  - 特定の条件を満たした時は後続のループ自体を処理したくない時に利用
- continue文
  - その時点でループの先頭に戻る
  - 特定の条件を満たした時はループ内の後続の処理をしたくない時に利用

関数

- 処理をひとまとめにしたもの
  - 外からは任意で引数を渡すことが可能
  - 処理の結果は必ず返さなくても良い
  - 複数の結果を返すこともできる
- うまく使えばプログラムの見通しが良くなり、再利用性が高まる

関数の定義

- defキーワードで関数を定義
  - 変数のスコープは関数内で閉じる
  - そのため、同名の変数を複数の関数で利用することができる

```
def add(x, y):
        return x + y

add(1,2)
>>> 3
```

演習問題1

- FizzBuzz
  - 1から100の数字を表示する関数FizzBuzzを定義しなさい
  - - 但し、FizzBuzz関数は3の倍数の時は数字のかわりに"Fizz"、5の倍数の時は数字のかわりに"Buzz", 3と5の倍数の時は"FizzBuzz"と表示しなさい
  - - FizzBuzz関数を実行し、期待通りの挙動になっていることを示すこと

演習問題2

- 九九の表示
  - 九九を表示するプログラムを書きなさい
  - 但し、一行に一つの段の結果を全て表示すること（9x9の九九の表が表示される）

演習問題3

- フィボナッチ数列
  - 10個のフィボナッチ数を求めて表示するプログラムを書きなさい
  - フィボナッチ数とは、前の２つの数字を加えると次の数になる、数列です。（ただし、1番目と2番目は1）
  - つまり、1, 1, 2, 3, 5, 8, 13 ....のように表示するプログラム

演習問題4

- フィボナッチ数列2
  - フィボナッチ数列を関数化して、任意の個数のフィボナッチ数を返すように修正しなさい

第4回： Python応用

Pythonの応用を学ぶ

アジェンダ

- Pythonの応用文法
  - リスト型
  - タプル型
  - 集合型
  - 辞書型
  - ファイルの入出力
  - モジュール

リスト型

リストは非常に重要なデータ型の１つ

- リストを操作するための様々な関数の紹介
- リストを使ったデータ構造の実現（キュー）
- リスト内包表記

```
>>> nums = [1, 2, 3, 4, 5]
>>> nums
[1, 2, 3, 4, 5]
```

リストを操作するための便利関数群

- リストの末尾に要素を追加：append()
- リストの要素で該当する最初の要素を削除：remove()
- リストの指定した位置に要素を挿入：insert()
- リストの順序を逆順に変更：reverse()
- リストの要素をソート：sort()

その他リストに対する様々な処理が関数として提供されている。

リスト内包表記

- シーケンス要素に対してある操作を行った結果をリスト化して返すリスト生成の特別な記法
- 非常に柔軟でかつ強力なため、覚えておく

```
# リスト内包表記によって0から5までの数字を生成
>>> nums = [i for range(6)]
>>> nums
[0, 1, 2, 3, 4, 5]
```

タプル型

- リストに似たデータ構造
  - スライスを利用した要素へのアクセスができる
- リストとの違いとして、immutableなデータ構造

```
# リストのように扱えるが、再代入などはできない
>>> nums = (1, 2, 3, 4, 5)
>>> nums[1]
2
>>> nums[1] = 0
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
```

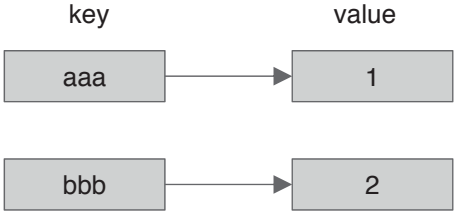
集合型

- 重複がなく、順序付けられていない要素をひとまとめにしたもの
- 様々な集合演算が扱える

```
>>> box = {'aaa', 'bbb', 'ccc', 'ddd'}
# 順序がないため、代入の時の順番が保証されているわけではない
>>> box
{'bbb', 'ddd', 'ccc', 'aaa'}
```

辞書型

- 辞書型は多言語では「連想配列」や「ハッシュ」と呼ばれているものに相当するデータ型
- キー(Key)と値(Value)の組み合わせを使うデータ型
  - Pythonのデータ構造のんかでも重要なものとなる



辞書型の基本

```
# 例えば、本のタイトルと値段とする
>>> book = {'aaa': 500, 'bbb': 1000, 'ccc': 10000}
>>> book
{'bbb': 1000, 'ccc': 10000, 'aaa': 500}
>>> book['aaa']
500

# 新しいkey-valueペアの登録
>>> book['ddd'] = 50
>>> book
{'bbb': 1000, 'ddd': 50, 'ccc': 10000, 'aaa': 500}
```

ループ処理との組み合わせ

- ループ処理に組み合わせると、key, valueのペアを順次処理できる

```
>>> book = {'aaa': 100, 'bbb': 200, 'ccc': 300}
# items()という関数でkey-valueペアを返す
>>> for key, value in book.items():
...     key, value
...
('bbb', 200)
('ccc', 300)
('aaa', 100)
```

ファイルの入出力：共通の流れ

ファイル処理の一連の流れ

- ファイルオブジェクトを作成
  - この時、モードを指定する
- モードに応じたメソッドを使ってファイルオブジェクトを経由した処理を行う
- 処理が終わったらファイルオブジェクトを閉じる

ファイルの入出力：書き込み

```
#open()によりファイルを開きファイルオブジェクトを取得する
# 'w'は書き込みモード
# 'r'は読み出し専用モード
>>> f = open('./tmp', 'w')

# writeメソッドで書き込む
>>> f.write("This is a pen.¥n")
>>> f.write("That is a cup of coffee.¥n")

# ファイルオブジェクトを閉じる
>>> f.close()
```

ファイルの入出力：読み出し

```
# 先程書き込んだファイルを再びopen()で開く
f = open('./tmp', 'r')

# ループで一行ずつ読み出せる
for line in f:
    print(line)

# 処理が終わったらファイルを閉じる
f.close()
```

モジュールとは？

- モジュールとは、再利用可能なプログラムの部品のようなもの
- Pythonには標準モジュールとして、各種の便利な機能が提供されている
  - 数学関連
  - 日付、時間関連
  - 文字列のフォーマッティング関連
  - etc...

モジュールのインポート

```
# import文を使ってモジュールをロードする
>>> import math

# dir()を使うとインポートしたモジュールで
# 利用可能な関数や定数の名前がわかる
>>> dir(math)
# すごく色々表示されるので省略

>>> math.pi
3.141592653589793
```

演習問題1

aからbまでの整数値がランダムで要素として格納されているリストを作成する関数generate_random_list()を作成しなさい。なお、以下の条件を満たすこと。

- 引数にリストの長さを与えることができる
- ランダムな値を生成する方法はいくつかある
  - random.randint(a,b)は a <= n <= b である整数nを返す

```
>>> import random
>>> random.randint(0,5)
0
>>> random.randint(0,5)
3
```

演習問題2

問題1で作成した関数を利用して、要素数100で0から10までの整数値が要素となっているリストを作成しなさい。そのリスト内の要素をチェックし、各整数値が何回ずつ出現したかをカウントする関数を作成しなさい

- 関数の引数は、問題1で作成した関数によって生成される要素がランダムなリスト
- ランダムなリストは、要素数100で、要素は0から10の整数値
- keyにリストの要素である整数値、valueにリストの要素が何回出現したかの値を持つ辞書が返り値

第5回： Numpyの基本

データ分析を行う上での必須ツールである
Numpyを学ぶ

アジェンダ

- Numpy基礎
  - Numpyとは？
  - Numpyのデータ構造：ndarray
  - ndarrayに関する基本的な操作
- Pandas基礎
  - Pandasとは？
  - Pandasのデータ構造：Series, DataFrame
  - Pandasに関する基本的な操作

NumPy（第2回目の講義より）

- NumPyはPythonのライブラリの一つ
- 大規模な多次元配列や行列の操作をサポート
- 特に行列演算が必要となる領域（画像処理）などで重宝されている
    - 深層学習も行列計算が必要になるため、必須ライブラリの一つ

NumPyとは？

- Pythonにおいて数値計算を効率的に行うためのライブラリである
- 型付きの多次元配列（numpy.ndarray）をサポートしている
- 多数の数学関数ライブラリを提供している
- 各種のデータ分析のライブラリで内部的に利用されていることが多い
    - Numpyそのものを使うことは少ないかもしれないが、各種ライブライブラリの動作の基本となっているため、理解しておくことで全体の理解度向上につながる

NumPyの基本的なデータ構造：ndarray

下記、「次元」という用語を使うが、配列構造の複雑さを指す

- 1次元配列：Pythonのリストと同じく、要素を羅列した構造
- 2次元配列：行列、リストの各要素がリストであるような構造
- 3次元配列：リストの各要素が2次元配列であるような構造
- etc...

と言う形式で、N次元まで配列構造を拡張することができる

ndarrayの生成

- Pythonのリストにarray()を適用することでndarrayは生成される
- リストをネストさせると次元の大きな配列が生成される
- array()以外にもndarrayを生成させる方法はいくつかある

```
# NumPyのarray関数を用いたndarrayの生成
>>> data1 = [6, 7.5, 8, 0, 1]
>>> arr1 = np.array(data1)
>>> arr1
array([6., 7.5, 8., 0., 1.])
```

ndarrayの属性

- ndarrayは属性としてshapeとdtypeを持つ
- shape
  - 配列の次元とそのサイズを格納するタプル
- dtype
  - 配列のデータ型を表す
  - 型としては、int32やfloat64など様々ななものがある
  - astypeを使うとデータ側の変換も行うことができる

```
>>> arr1.shape
(5,)
>>> arr1.dtype
dtype('int64')
```

ndarrayとスカラーの計算

- ndarrayの演算の基本ルール
  - 同じサイズのndarray同士の算術演算は、同位置の要素同士で計算される
- 「スカラー」とは、単なる数字のこと
- ndarryaとスカラーの計算ルール
  - ndarrayの全ての要素に対して、スカラーとの演算が適用される

インデックス参照とスライシング

- ndarrayはPythonのリストのようにインデックスやスライスで参照することができる
- 参照によって切り出されたndarrayは全て元のndarrayに対する参照である
  - 参照した部分ndarrayに対する変更は元のndarrayに対する変更となる

ブールインデックス参照

- 参照に論理演算を使うことができる
  - 条件に合致している要素はTrue
  - 条件に合致していない要素はFalse

  がリストとして返ってくる

- そのTrue, Falseのリストをさらに参照に利用すると、Trueの要素のみ取り出すことができる

行列演算

- 標準的な行列に関する種々の演算を提供している
  - 内積
  - 行列の分解
  - etc...
- numpy.linalgモジュールで定義されている

ユニバーサル関数

- ndarrayの要素全てを対象に処理を適用させる関数
  - 各要素の平方根を取得するsqrt()
  - 各要素の指数関数を取得するexp()
  - etc...

数学関数、統計関数

- ndarrayの全体、あるいは特定の軸（行方向、列方向）に対して適用できる関数群
  - ndarray全体に対して適用する数学関数の例
    - 平均値を求めるmean()
    - 和を求めるsum()
  - ndarrayの特定の軸に対して適用する数学関数の例
    - 累積和を求めるcumsum()
      - cumsum(0)は行方向、cumsum(1)は列方向の累積和を求める

ソート、集合関数

- 標準的なリストと同じくソートもできる
- ただし、多次元配列であるため、数学関数と同じく、ソートの軸を定める必要がある

- また、PythonのSetと同じく集合関数も定義されている

乱数の生成

- サンプルデータ生成において非常に重要な機能の1つ
- 様々な統計的な分布に基づいた乱数を要素に持ったndarrayを生成することができるため、大変重宝する

演習問題1

- 1次元のndarrayであるarrを作成せよ
  - 要素は[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
- arrにユニバーサル関数である平方根を適用せよ
- ブールインデックス参照を用いて平方根を適用する前の元のarrから偶数の要素のみ抜き出せ

演習問題2

- 要素が1, 2, 3, 4のndarrayを用いて2x2の2次元のndarrayであるarr2を作成せよ
- arr2を転値させて2倍した新しい2次元のndarrayであるarr3を作成せよ
- arr3とarr2の和と積を計算せよ

第6回： Pandasの基本

データ分析を行う上での必須ツールである
Pandasを学ぶ

アジェンダ

- Pandasとは？
- Pandasのデータ構造
  - Series
  - DataFrame
- Pandasの基本的な使い方

Pandas

- PandasはPythonのライブラリの一つ
- データ分析を支援するための機能を多数提供している
- DataFrameと呼ばれるテーブル形式のデータフォーマットを提供
  - SQLライクなデータ操作を宣言的に記述できる
  - DataFrameと外部データソース間で相互に読み書き可能
- 時系列データもサポート

Pandasとは？

- データ分析を行う際の様々な便利機能が提供されているライブラリ。データ分析を行う上では必須のライブラリと言える
- 特にDataFramと呼ばれる表形式のデータ構造は非常に強力である
- Pandasは非常に豊富な機能を持っている
- 今回の講義では、ファイルからデータを読み込み、簡単な操作をする流れを見る

Pandasの基本的なデータ構造

- Series
  - 一次元の配列のようなもの
  - 順序を持っている
  - 名前の通り、時系列データを扱うのが得意
  - データにラベルを持たせることができる

- DataFrame
  - 二次元のテーブルのような構造
  - 単純な二次元のリストではなく、行と列に対して「ラベル」を付与することができる
  - Excelのようなデータ形式をイメージすれば良い
  - Pandasと言えばDataFrameというくらいに非常に強力なデータ構造

Pandasの処理の前に

- 今回はプログラムの実行ファイルと同じディレクトリ内に”iris.csv”というデータファイルが置かれている前提とする
- “iris”はアヤメに関する有名なデータセット
  - 3種類のアヤメのデータ
  - がく片の長さ、がく片の幅、花片の長さ、花片の幅の4つの特徴量からなる
  - データの詳細自体は次回以降に詳細を解説する

データファイルからの読み込み

- 種々の読み込み用の関数が提供されているため、それを利用する
  - 今回は対象のファイルがCSVファイルのため、read_csv()メソッドを利用する
  - CSVファイルの状況に応じて、種々のオプションを使い分ける
    - ヘッダーがあるとかないとか
    - etc...

カラム名をつける

- colums関数にリストを渡すことでカラム名をつけることができる
- カラム名をつけることでデータへのアクセスがスムーズになる

列の抽出

- アクセスしたいカラム名をリストで渡すことで参照することができる
- 列全体を抽出し、1列のみの場合はSeriesとして、2列以上の時はDataFrameとしてデータが抽出される

行の抽出

- .ixを使うことで、Pythonの標準のリストのスライスのようにアクセスすることができる
- 行のアクセスに列のアクセスを組み合わせることで、特定の行と列を抽出することもできる

数値のソート

- sort_values()関数を使うとソート列の値でソートを行える
  - ソートの基準とするカラムをリストで指定できる
- 指定した順に昇順にソートさせるため、複数のリストを与えると列1でソートした結果をさらに列2でソート、というような結果が得られる

数学関数/統計関数

- 種々の数学や統計関数が提供されています。
- describe()は列ごとのよく使う統計量を算出してくれる便利な関数です。

- 各種関数は列方向にかけるのか、行方向にかけるのかを指定することができます。
  - 列方向にかけると、ある列を全てのデータについて集計することに相当
  - 行方向にかけると、全ての列をある行のデータについて集計することに相当
  - 覚えておくと大変便利な機能

ピボットテーブル

- Excelでよく利用されるピボットテーブルも関数pivot_table()で簡単に実現できる
- 集計したい属性を選択し、行と列を指定する
    - オプションで自由に行いたい処理を指定できるため、単純な集計以上のことも実現できる

Group by操作（グルーピング）

- グルーピングを行うことで、特定の要素に対して処理を適用することができるようになる
- グループ化の対象となる列を指定し、グループ化されたオブジェクトを作成する
    - グループ化したオブジェクトに集計の関数を適用することができる
    - 関数適用時には、特定の列を指定するなど、通常のDataFrameのようアクセスすることができる

演習問題

- irisデータに対しての更に処理を行うことで理解を深める

- SpeciesがsetosaとvirginicaのデータのSepalLengthとPetalLengthを抽出せよ
- 抽出した結果をPetalLengthが小さい順にソートして並び替えよ
- Species毎のレコード数と統計量を確認せよ

第7回： データの可視化

データの可視化を学ぶ

アジェンダ

- データの可視化とは？
- matplotlib
- グラフ描画の基本
  - 折れ線グラフ
  - 散布図
  - ヒストグラム
  - タイトルを表示
  - 表示幅の変更
- 複数のグラフの描画
  - グラフを別々に描画し、1つの図とする方法
- クラスを利用した描画方法
- 2つのグラフで軸を揃えて描画
- 凡例をつける方法

データの可視化とは？

データは単なる数値の羅列として見るだけでなく、可視化することで得られる知見が得られる可能性がある。

そのため、データの可視化はデータ分析のプロセスにおいて非常に重要なステップの１つといえる。

データに外れ値がある場合

- 一般的に、このような状況下では、平均値と中央値に大きな乖離が生じる
- 統計的な知識がある人にとっては、データに何らかの外れ値が含まれている可能性を考えることができる
- 一方、統計的な知識のない人にとっては、外れ値の可能性を見抜くことができない
- また、統計的な知識がある人でも、外れ値の可能性を見逃す可能性はある
- しかし、データを可視化すると、明らかに異常な位置にデータがあることが見て取れるため、外れ値の可能性を見抜くきっかけとなる

データを可視化する上で必要なこと

- どんなデータを対象に可視化を行うか
- 何のために可視化を行うか
- どんなグラフを使って可視化を行うか

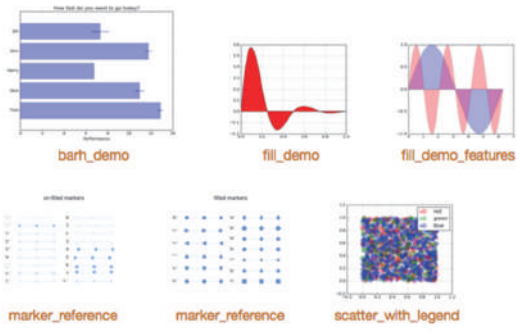
という三点が非常に重要である。

本講義では、その中でも「どんなグラフを使って」の部分にフォーカスを当て、様々なグラフをPythonによって描画できるようになることを目的とする

本講義では、matplotlib.pyplotを中心にしてグラフ描画を行う

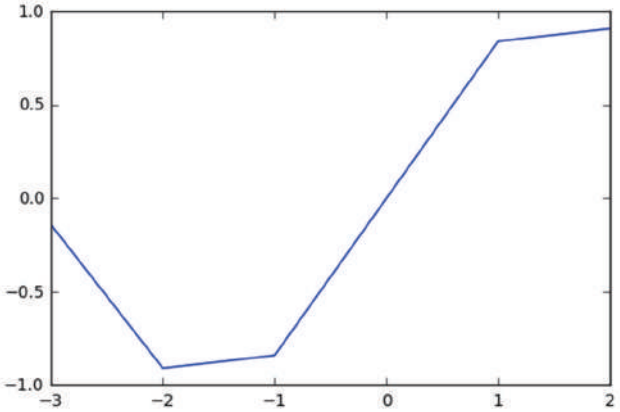
matplotlib

- matplotlibはPythonのライブラリの一つ
- グラフの描画機能を提供している
  - 描画したグラフを多数の画像形式に変換可能
- Pythonにも様々なグラフ描画ライブラリはあるが最も普及しているものの一つ



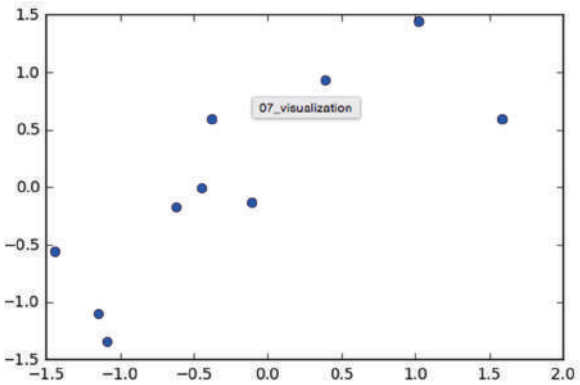
グラフ描画の基本（折れ線グラフ）

- 折れ線グラフとは、各データを線で繋いだもの
  - numpyなどでx, yのデータを適当に生成する
  - plot()関数を使ってグラフを描画



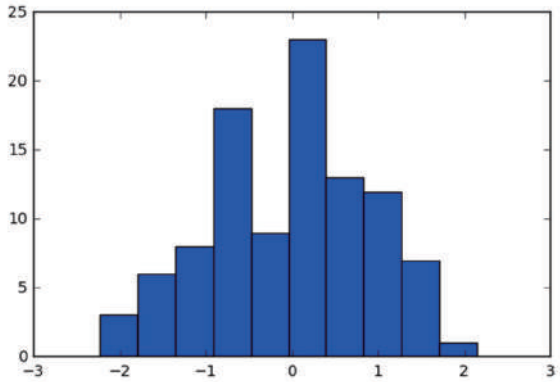
グラフ描画の基本（散布図）

- 散布図とは、データを点でプロットしたもの
  - 折れ線グラフと同じくplot()関数で描画する
  - plot()関数に与えるオプションで散布図を描画する



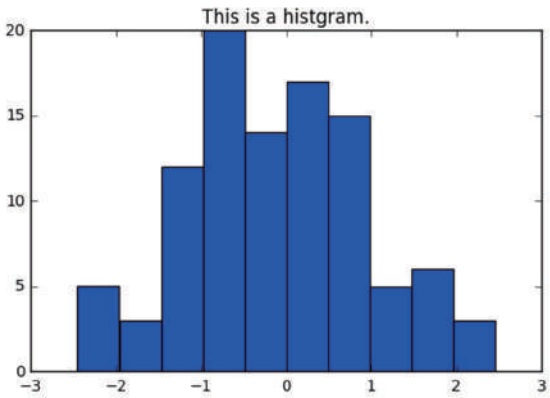
グラフ描画の基本（ヒストグラム）

- ヒストグラムはあるデータがどのくらい集まっているかカウントして表示するためのグラフである
  - ヒストグラムの描画にはhist()関数を利用する



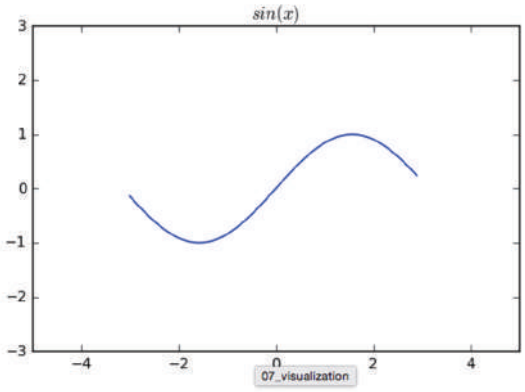
グラフにタイトルを表示

- グラフにはタイトルをつけた方がわかりやすい
- title()関数を使うことでタイトルをつけることができる



グラフの表示幅を変更

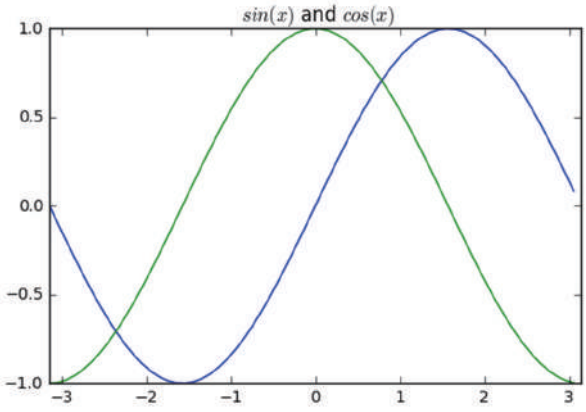
- plot()関数は与えられたデータ点に合わせて自動で表示幅を調整する
- グラフによっては自分で表示幅を調整した方が視認性が高まる場合がある



表示幅を変更した例
ただし、これはあまり
視認性の向上には役立っ
ていない。

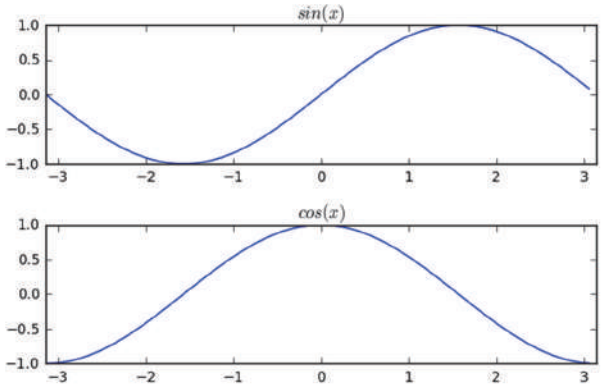
複数のグラフの描画

- 複数の結果を１つのグラフに描画する例
  - 結果を比較したい時などでは非常に便利
  - plot()を2度呼び出せば描画できる



グラフを別々に描画し、1つの図とする

- 複数の結果を１つのグラフではなく、別々のグラフに描画し、かつ１つの図として取り扱いたい場合の例
  - subplot()を使い、描画するグラフの位置を指定しながらグラフを描画する



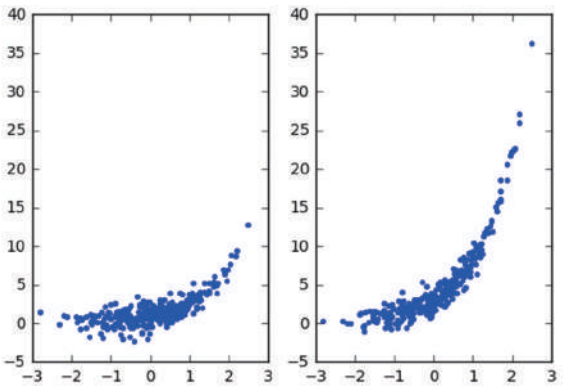
クラスを使う描画方法

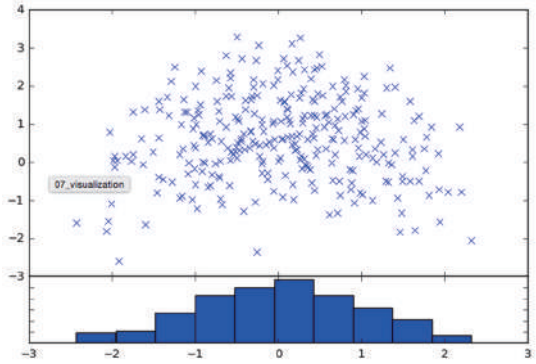
matplotlib.pyplotでは、関数の裏側で描画に利用する各種クラスを使っている。裏側で使われている各種クラスを理解することで、より柔軟に様々なグラフを描画できるようになる。

1. Figureインスタンスを生成する
2. Axesインスタンスを生成する
3. データを渡して、プロットを行う
4. y軸の範囲を調整し、タイトルやラベル付けなどを行う

2つのグラフで軸を揃えて描画

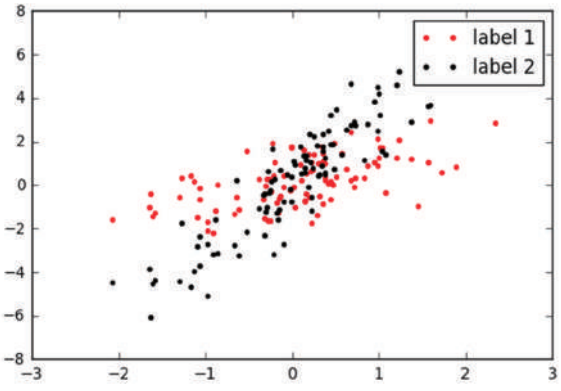
- 明示的に設定することで、2つのグラフの尺度を揃えることもできる
  - 2つの結果を同じ尺度で比較できるようになる
  - x軸を揃える方法、y軸を揃える方法がある
    - plot()のオプションで指定する





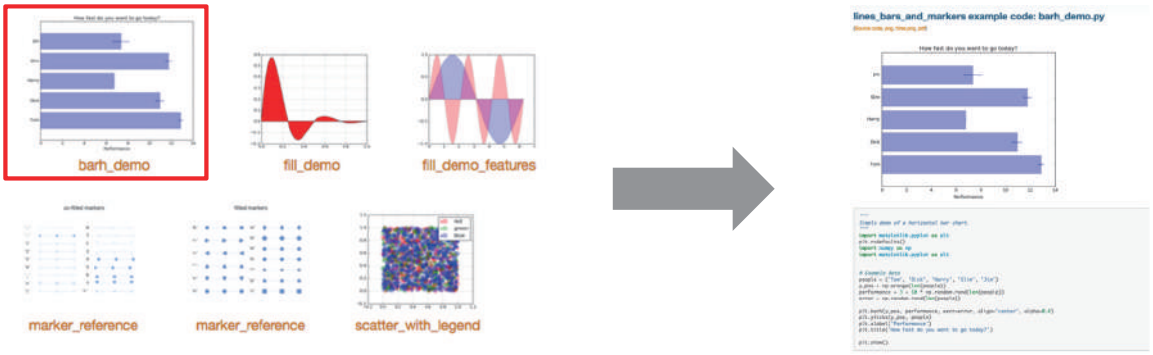
凡例をつける

- 凡例をつけることでグラフの意味がより伝わりやすくなる場合がある
  - plot()のオプションで指定することができる
  - 凡例の表示位置の指定もすることができる



さらに学ぶには？

1. matplotlibのGalleryのページを見て、自分の書きたいイメージに近いグラフを探します（http://matplotlib.org/gallery.html）
2. グラフをクリックするとソースコードを見ることができるので、ソースコードからグラフの書き方のヒントを得ます



演習問題

- 前項で紹介したmatplotlibのGalleryのページを確認せよ
- その中から１つ選び、身の回りにあるデータ、もしくは自分で生成したデータを使ってデータを可視化せよ
- データを可視化したことでわかりやすくなった点を説明せよ

第8回：統計解析全体像

統計解析の全体像を学び
この後の数回にわたる講義のガイドとする

アジェンダ

- 統計解析とは？
- データから「事実を正しく語る」ことについて
- 統計という道具を使う
  - データ収集
  - 記述統計
  - 推定
  - 仮説検定
  - 相関
  - 回帰
  - etc...

第1回より再掲

キーワード２：統計

Wikipediaの「統計」によると…

- 統計（とうけい、statistic）は、現象を調査することによって数量で把握すること、または、調査によって得られた数量データ(統計量)のことである。

第1回より再掲

つまり、統計とは・・・

現象をデータから読み解くための

技術やその結果そのもの

統計学はその技術を

体系立てた学問

第1回より再掲

なぜ統計が必要か？

- データは簡単に、かつたくさん手に入るようになった

- 次に必要なのは、データから「何が起きているのか」を正確に理解し、「次の行動の意思決定材料」とする方法
- 手元にあるデータを正しく理解するために道具としての統計が必要となってくる

第1回より再掲

統計学について

Wikipediaの「統計学」によると...

- 統計学は、経験的に得られたバラツキのあるデータから、応用数学の手法を用いて数値上の性質や規則性あるいは不規則性を見いだす。統計的手法は、実験計画、データの要約や解釈を行う上での根拠を提供する学問であり、幅広い分野で応用されている。 現在では、医学（疫学、EBM）、薬学、経済学、社会学、心理学、言語学など、自然科学・社会科学・人文科学の実証分析を伴う分野について、必須の学問となっている。また、統計学は哲学の一分科である科学哲学においても重要なひとつのトピックスになっている。

第1回より再掲

つまり、統計学とは・・・

データを正しく解釈し、意思決定を

行うために必須の学問

これまでは学問領域での適用に

とどまってきたが、ビジネス活動にも

必要とされるようになり始めている

第1回より再掲

統計解析とは？

- データサイエンティスト、ビッグデータ、統計、機械学習、AIなど、データ分析に関わる種々のキーワードを中心に説明を行い、業界イメージをつけるとともに、講義全体（全15回）の全体像を理解する
- また、講義内容をもとにして自由課題の演習を実施する

データから「事実を正しく語る」とは？

- ある事象に対して何らかの主張をしたい場合、どうすれば「正しく」主張することができるだろうか？

事象：第一子は第二子よりも身体的に大きな子に育ちやすい

もちろん、医学的に立証された正しい説ではありません。思考実験です。

事象に対しての主張方法

- 「私には子供が2人いて、第一子よりも第二子の方が大きい。なのでこの主張は正しい」「また、近所の家の子も同じである。この説は信憑性が高い」
  - これは本当に正しいと言える？

- 「私の姉には子供が3人いて、第二子が一番体が大きい。なので、この主張は正しくない」
  - この反証も正しいといえるだろうか？

この主張が正しいと言い切れない理由

- 標本数が小さすぎる
  - 「標本」とは観測されたデータのことを言う
  - そもそも標本数1や2では「たまたま」の可能性を排除しきれない
- 選択バイアス
  - そのそもこういった議論に参加する人は、この説に興味を持っている人
  - なので、主張に当てはまる事象だけを意図的に集めている可能性がある
- 確証バイアス
  - この主張が「正しい/正しくない」という前提に立っているため、公平な主張ではない
- 不正確さ
  - そもそも定性的であり「何を持って身体が大きい」かの定義がない状態で主張が行われている

統計という道具を使う

- そこで、データから客観的な事実を語るための道具として、統計を使う
- また、さらに深掘りするために統計解析を行う

- 統計解析は以下のような要素からなる
  - データ収集
  - 記述統計
  - 推定
  - 仮説検定
  - 探索的なデータ解析
    - 外れ値チェック
    - 相関チェック
    - 回帰
    - etc...

データ収集

- まずは分析に必要なデータを集める
  - データがないと分析自体が行えない
- 良質なデータを多く集めることが重要
  - が、実際は良質なデータほど多くのデータを集めることは難しい

- 以後、Pythonのscikit-learnに同梱されているサンプルデータであるirisを例にして話を進めて行く

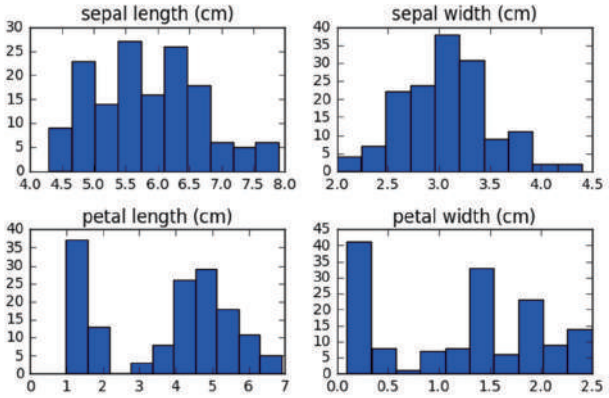
記述統計

- データを集計しその特性について簡潔に理解しやすくする方法

- ヒストグラム（度数分布）
- 平均値、中央値
- 分散、標準偏差
- 四分位数
- etc..

ヒストグラム（度数分布表）

- データのばらつきを確認するための方法
- １つの変数毎に値を見ていく



irisデータの場合、変数が4つあるため、それぞれでヒストグラムを確認

変数毎にデータのばらつきが異なることが分かる

平均値、中央値、分散、標準偏差、四分位数

- pandasの関数で各変数について一気に確認することができる

・レコード数
・平均値
・標準偏差
・最小値
・四分位数(25%)
・中央値
・四分位数(75%)
・最大値

の順で並んでいる

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

平均値の意味

- 平均値
  - データの総和をデータの個数で割ったもの
  - データ１つあたりの数値を表す

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i = \frac{x_1 + x_2 + \cdots + x_n}{n}$$

nはデータの個数：irisの場合は150
xは1つ１つのデータの値：irisの場合、１つのデータのとある変数

数式の出典：https://ja.wikipedia.org/wiki/%E5%B9%B3%E5%9D%87

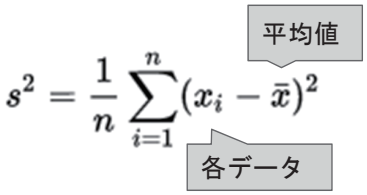
中央値の意味

- 値を小さいものから順に並べ直す
- ちょうど真ん中の値が中央値となる
  - irisの場合だと、データ数が150個のため、小さいもの順に並べた75番目の数字

- 平均値と中央値の関係
  - 平均値の意味は「代表的によく現れる数値」とも言える
  - 代表的によく現れる数値が「真ん中」に表れている場合、平均値と中央値は近しい値となる
  - 逆に言うと、平均値は必ず「真ん中」の数字ではないことに注意する必要がある

分散

- 分散
  - データがどの程度散らばっているか、を表したもの
  - 平均値と各データの差の二乗を、全てのデータについて足し合わせたもの



二乗を取る理由としては、平均値より大きい値は正、小さい値は負になってしまい、打ち消し合う可能性があるため。
二乗を取ることで必ず正の値とし、尺度として使えるようにする
※意図的に二乗をとる数学的なテクニック

数式の出典：https://ja.wikipedia.org/wiki/%E5%88%86%E6%95%A3

標準偏差

- 分散に対して平方根を取ったもの
  - 分散は二乗しているため、もとのデータと単位が異なる
    - 分散を見ているだけでは、実際のデータではどういう意味を持つのかがイメージしにくい
  - 平方根を取ることで、元のデータと単位をそろえた、ばらつきの尺度

- 直感的に理解されやすいという観点で、ばらつきの尺度としては、標準偏差の方が採用されやすい

母集団、標本

- 母集団
  - 調査対象になっているデータの全体を表す
- 標本
  - 調査対象になっているデータから、実際に取得されて手元にあるデータ

- irisを例にすると
  - 世の中にある三種類のアヤメ（Setosa, Versicolor, Virginica）全体が母集団
  - 実際に全てのデータを入手することは不可能なためサンプルとして一部のデータしか得られない。それが標本
- 標本から母集団の性質を明らかにすることが統計解析の大きな目的の１つ

演習問題

- 身のまわりで平均値と中央値が一致しないような例を探してみよ
  - なぜ平均値と中央値が一致しないのかその理由を述べよ

第9回：推定、検定

推定、検定を学ぶ

アジェンダ

- 標本抽出
- 区間推定と信頼区間
  - 母平均算出の例
- 検定、確率分布
  - t検定
  - カイ二乗分布
- 演習問題

標本から母集団の推定

- あるデータを得た時、それが母集団であることは滅多にない
  - たいていは標本である
- 限られたデータである標本から、データ全体である母集団の特徴を類推する方法として「推定」がある

区間推定と信頼区間

- 標本統計量から母集団統計量の値が含まれるであろう「区間」を推定すること
- 母集団の平均（母平均）を求めるには以下の情報が必要
  - 標本平均
  - 不偏分散
  - 推定の誤差をどの程度まで抑えるか
    - 信頼区間と呼ばれる

- 信頼区間は一般的に95%で設定されることが多い
  - つまり統計的に5%は信頼区間の幅を越える可能性があるという推定

不偏分散

- 不偏分散とは？
  - 標本の分散は母集団の分散の推定値にはならないことが統計的に証明されているため、不偏分散と呼ばれるものを利用する
  - 不偏分散は、分散を求める時に利用した、分母のデータの個数を1つ減らすだけ

$$\frac{1}{n-1}\sum_{i=1}^{n}(X_i-\overline{X})^2$$

通常の分散を求める時は
nとなっている所

数式の出典：http://www.sist.ac.jp/~kanakubo/research/statistic/fuhenbunsan.html

母平均算出の例

- 母平均とは母集団の平均値のこと
- サンプルデータである標本から、データ全体の平均を見極めることに相当

- 母集団の分散が分かっている時と、分かっていない時で少し方法が異なる
  - 今回は母集団の分散が分かっていない時を取り上げる

母分散が分かっていない時の母平均の推定（1/2）

- 設定したい信頼区間に合わせて、分布の統計量を決める必要がある。分布の統計量値は分布表と呼ばれるものから必要な値を決める。
  - 標本が大きな時は正規分布
  - 標本が小さな時はt分布
  - 決めるのに必要な情報は、自由度（標本数-1）、信頼区間の２つ

- 分布表は統計の教科書やWebなどで公開されている
  - http://www.biwako.shiga-u.ac.jp/sensei/mnaka/ut/tdistinvtab.html
- 例えば、標本数が15で、95%の信頼区間に対応する統計量を探すと、2.145となる

母分散が分かっていない時の母平均の推定（2/2）

- 信頼区間の上限値と下限値を求める
- 下限値
  - (標本平均) − (分布の統計量) × (不偏分散の平方根) ÷ (標本数-1の平方根)
- 上限値
  - (標本平均) + (分布の値統計量) × (不偏分散の平方根) ÷ (標本数-1の平方根)

上限値と下限値に挟まれた間に母集団の平均値があると想定できる

母平均推定の例題

- 全国の高校生1000人を対象に100点満点の数学のテストを行った
- テストを受けた学生から無作為に選んだ10人の点数は下記の通りだった
  - 65, 58, 40, 80, 45, 50, 85, 40, 95, 80
- この時、母平均の信頼度95%の信頼区間はいくらか？

解答

- 標本平均：63.80
- 標本標準偏差：19.13
- 分布の統計量：2.26（信頼区間95%、t分布、標本数9の統計量）

となるため、信頼区間は 49.37 < 母平均 < 78.23 に存在していることになる。

検定

- 得られているデータは標本であるため、標本のデータ間で差異があるものが母集団に対しても差異があることを主張するための方法
- 考え方としては次の通り
  - 主張したい仮説と逆の仮説（帰無仮説）を立て、それが統計的に棄却させる
  - 帰無仮説を棄却することで、反対の仮説（対立仮説）が成立する

- つまり、「新しい薬に効果がある」ということを主張したい場合
  - 「新しい薬は効果がない」を帰無仮説とする
  - 新しい薬、古い薬に関しての差を統計的な検定にかける
  - 新しい薬と古い薬に効果がない、ということは統計的に滅多に起こらないことであることを証明し、帰無仮説を棄却する
  - 帰無仮説が棄却されたため、対立仮説である「新しい薬には効果がある」が成立する

仮説検定の例：t検定

- 検定の種類の１つ
- 2組の標本についての平均に有意差があるかどうかの確認に用いられる手法
- t検定の行い方
  - 前提：帰無仮説は「2組の標本の平均に差は無い」
  - 2組の標本の差の平均を求める
  - 2組の標本の差の標準偏差（不偏分散の平方根）を求める
  - t分布の統計量を求める
    - (標本の差の平均) / ((標本の差の標準偏差) / (標本数の平方根))
  - 条件に合致するt分布表の統計量を比較し、標本の差から得られた統計量の方が大きければ、帰無仮説は棄却される
    - つまり2組の標本のには差がある、となる

t検定の例題

- とある高校で2年続けて同じ人を対象に100点満点の英語のテストを行った
  - 2015年の結果：70, 60, 60, 80, 90, 55, 85, 60, 95, 80, 70, 90, 80, 60, 70
  - 2016年の結果：70, 50, 55, 75, 50, 40, 90, 55, 80, 70, 60, 80, 85, 60, 80
- 2年間の平均点には差があると言えるか？
- 平均点は下記の通り
  - 2015年：73.67
  - 2016年：66.67

解答

- 帰無仮説：2年間のテストの点数の平均に差はない
- 標本の差の平均：7.00
- 標本の差の標準偏差：11.37
- 標本の差から得られた統計量：2.30
- 信頼区間95%、標本数14のt分布の統計量：1.76

標本の差から得られた統計量が、分布表から得られる統計量を上回っているため、帰無仮説は棄却され、対立仮説である「2年間のテストの点数の平均に差がある」が採択される

仮説検定の例：カイ二乗検定

- 仮説検定の一手法
- 観測されたデータ（標本）の分布は、理論上の分布とほぼ同じと見なせるかを確認するための手法
- カイ二乗検定の行い方
  - 前提：帰無仮説は、観測されたデータ（標本）の分布は、理論上の分布と一致する
  - 項目ごとの観測値と理論値をまとめる
  - ((観測値) - (期待値))の2乗 / (期待値)　の値を求める（これがカイ二乗値となる）
  - 全ての観測値のカイ二乗値を足し合わせる（これが今回比較対象となる統計量）
  - (項目の数) - 1 が自由度となる
  - カイ二乗分布から、条件に合う統計量を探す
  - 算出した統計量と、条件に合致する分布表から見つかる統計量を比較する
  - 算出した統計量の方が大きければ帰無仮説が棄却される

カイ二乗分布の例題

- サイコロは歪みがないものであれば、120回サイコロを降ると出目の確率は全ての目で同じ20回になるのが理想的である
- とあるサイコロを120回振った結果、以下のような結果となった
  - 1の目が出た回数：25
  - 2の目が出た回数：27
  - 3の目が出た回数：22
  - 4の目が出た回数：10
  - 5の目が出た回数：11
  - 6の目が出た回数：25
- 今回このような結果が出たサイコロは、歪みのないサイコロといえるか？

解答

- 標本から得られた各出目のカイ二乗値の和：14.20
- 信頼区間95%、自由度5のカイ二乗分布の統計量：11.07

- 標本から得られた値の方が大きくなったため、帰無仮説である「サイコロに歪みがない」は棄却された。よって対立仮説である「サイコロに歪みがある」が採択される

演習問題

- 今、日本全体の人口は男女比が1:1であるとする
- ある地域の交通量調査を行ったところ、100人の通行者の割合は男：59人、女：41人となった
- この結果は、日本全体と同じような男女比率を元にした調査結果であると言えるか？

参考：検定チートシート

説明変数(x)	目的変数(y)	検定方法
連続値	連続値	相関
カテゴリカルな値で2つのグループ	連続値	t検定
カテゴリカルな値	カテゴリカルな値	カイ二乗検定

第10回：相関

相関について学ぶ

アジェンダ

- 相関について
- 相関関係と因果関係
- 2変数の相関関係
  - 相関係数
  - 散布図によるプロット
- 2変数以上の相関関係
  - 相関行列

相関について

- 2つの変数の間の関係性について数値で記述した分析手法
- 一方が変化すればそれに応じて他方も変化する、という関係
  - Aが上がれば、Bも上がる
  - Bが上がれば、Aも上がる

- 似ている概念に因果関係があり、それとは異なるので注意が必要
  - 因果関係もAが上がれば、それが原因でBが上がる、という関係
  - ただし、Bが上がればAが上がるとなはならないため、一方向の関係

2変数の相関関係：相関係数

- 相関係数は、2つの変数に相関があるかないか、またその強さについて表した指標
- -1から1までの値を取る
  - 0：無相関
    - そもそも２つの変数に相関関係はない
  - 0から1：正の相関
    - 2つの変数に正の相関がある。正の相関とは、片方の変数が大きくなるともう一方の変数も大きくなる、ということ
  - 0から-1：負の相関
    - 2つの変数に負の相関がある。負の相関とは、片方の変数が大きくなるともう一方の変数は小さくなる、ということ

相関係数の求め方

- (相関係数) = (共分散) ÷ (変数Aの標準偏差 × 変数Bの標準偏差)
- 共分散は、下記の値を全ての観測値で足し合わして観測数で割ったもの
  - (変数Aのとある観測値 - 変数Aの平均)×(変数Bのとある観測値 - 変数Bの平均)

- すなわち、以下の情報があれば相関係数を求めることができる
  - 変数Aの平均と標準偏差
  - 変数Bの平均と標準偏差

相関係数を求めるの例題

- irisのsepal length(cm)（がく片の長さ）とpetal length(cm)（花弁の長さ）の相関係数を求めよ

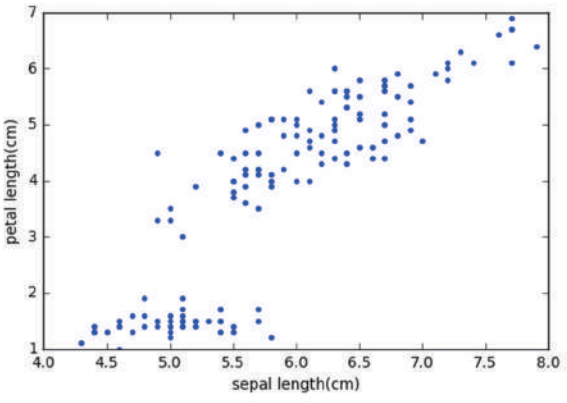
解答

- 共分散は1.27
- 相関係数は0.87
  - 非常に強い正の相関があるといえる

- なお、Pandasには変数間の共分散、相関係数を求めてくれる関数も用意されている
  - cov()関数
  - corr()関数

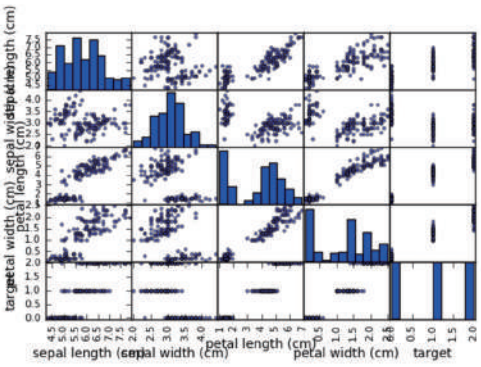
2変数の相関関係：散布図によるプロット

- irisの’sepal length(cm)’, ‘petal length(cm)’ の関係を散布図としてプロット
- 正の相関が強い変数のため、右肩上がりとなっている
  - 逆に負の相関だと右肩下がり、無相関だと方向性がなくなる



2変数以上の相関関係：相関行列

- 変数ごとの相関関係を一気に見るものを相関行列と呼ぶ
  - pandasには実現する関数が提供されている
  - 対角線がヒストグラム
  - それぞれの変数の掛け合わせでの散布図が描かれる



演習問題

- 例題ではirisの種類自体には着目していなかったので、種類にも注目して相関関係を調べる

- irisのSetosaとVersicolorのがく片の幅と花弁の幅の相関係数を求めよ。また散布図を図示せよ。
- irisのVersicolorとVirginicaのがく片の長さと花弁の幅の相関係数を求めよ。また散布図を図示せよ。

第11回：回帰分析

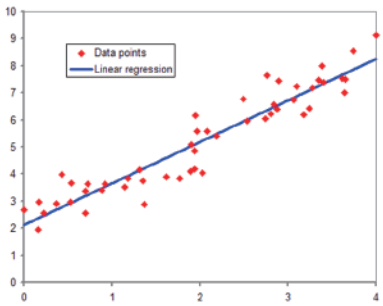
回帰分析を学ぶ

アジェンダ

- 回帰 / 回帰分析とは？
- 線形回帰、非線形回帰
- 最小二乗法
- Pythonによる最小二乗法の実行
- 最小二乗法による回帰の例
  - 線形回帰
  - 非線形回帰
- 演習問題

回帰 / 回帰分析とは？

- 説明変数を使って目的変数を説明するモデル（関係）を探すことを回帰と呼ぶ
- 一般的に説明変数はx、目的変数はyとなる
- 統計的な手法で回帰を行うことを回帰分析と呼ぶ



赤い点が与えられた時、関係性を表すために、
一番シンプルにする場合は、青い直線を
引けば良い。これが回帰分析である

例えば、直線は１次関数　$y = ax + b$　にて
描くことができる。
すなわち、aとbを求めることに相当する

出典：https://upload.wikimedia.org/wikipedia/commons/b/be/Normdist_regression.png

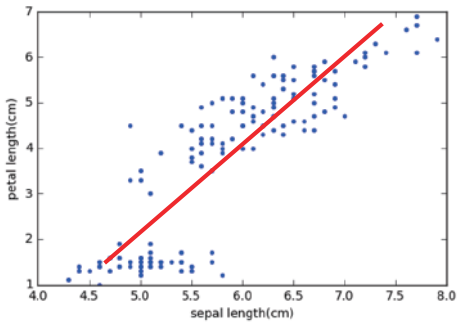
線形回帰、非線形回帰

- 線形、すなわち直線で関係性を表すことを線形回帰と呼ぶ
  - 目的変数と説明変数が直線で表せるような関係の場合
  - 直線だけで当てはめることができない関係も多い

- 非線形、すなわち直線ではない形で関係性を表すことを非線形回帰と呼ぶ
  - より複雑な関係を表現していることになる
  - 複雑かつ広範な範囲のため講義では取り扱わない

線形回帰について

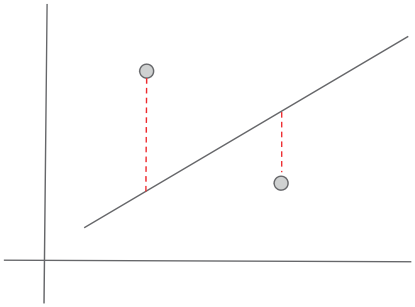
- 例えば前回の講義で相関があると確認された、irisの”petal length”と”sepal length”について考える
  - 右肩上がりの関係があるのが確認でき、直線を引くことができそうというのが分かる
  - 直線の傾き、及び切片を求めることが回帰分析に相当する
- 説明変数がpetal length、目的変数がpetal lengthである



より適切な直線の傾きと切片を求めるのが線形回帰の目的

最小二乗法

- 最小二乗法は偏微分などのやや高度な数学的な話が出てくるため、ここではイメージの説明のみとする
- 端的に言うと、直線とデータ点の差が最も小さくなるような傾きと切片を求めることに相当する
  - 各データ点と直線の差を足し合わせ、それが最小となるような傾きを決める



赤い点線の合計が最も小さくなるように直線を引く

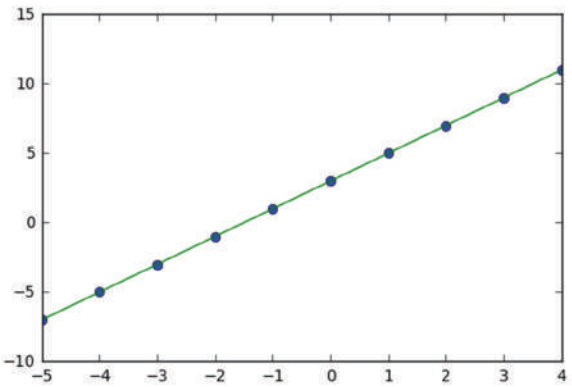
Pythonによる最小二乗法

- Numpyに最小二乗法を行ってくれる関数が用意されているため、それを利用する
- numpy.polyfit
  - また、その後のグラフ描画に便利なpoly1dも提供されているため、非常に便利

- まずはじめに関数が分かっている状態で最小二乗法を試してみる
  - y = 2x + 3
  - 最小二乗法を適用して、2と3に近しい値が求められることを確認する

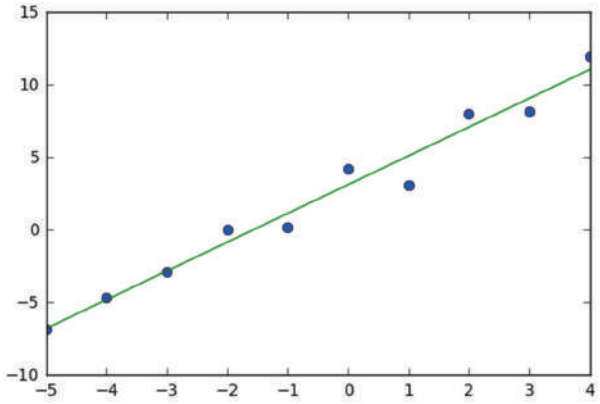
最小二乗法による回帰：１次関数（1/2）

- 線形関数を元にして作成したデータ点から、最小二乗法で直線を引くと確かに直線が引かれることが確認できる



最小二乗法による回帰：１次関数（2/2）

- 線形関数を元にして作成したデータ点にノイズを乗せ、より実践的な状態で最小二乗法を使う
- 確からしい直線が引かれることが確認できる

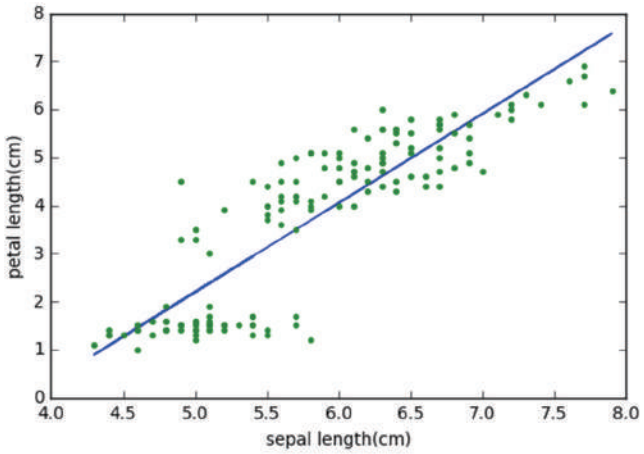


例題：irisデータに対しての最小二乗

- 前回の講義で確認したirisデータの「がく片の長さ」と「花片の長さ」の関係に対して、線形回帰を行い直線を引きなさい
  - 正の相関があるため、右肩上がりの直線が引くことができそうと予想できる

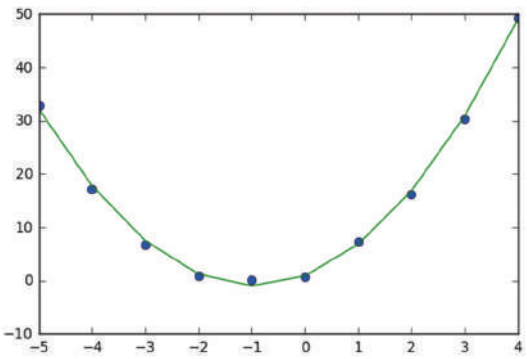
解答

- 確かに右肩あがりで、データ点の中央を通るような直線が引けることが確認できる



最小二乗法による回帰：2次関数

- numpy.polyfit()は2次関数以上（非線形）に対しても、最小二乗法を行うことができる
  - 関数が2次関数以上になると、直線ではなくなる。つまり非線形な関数となる

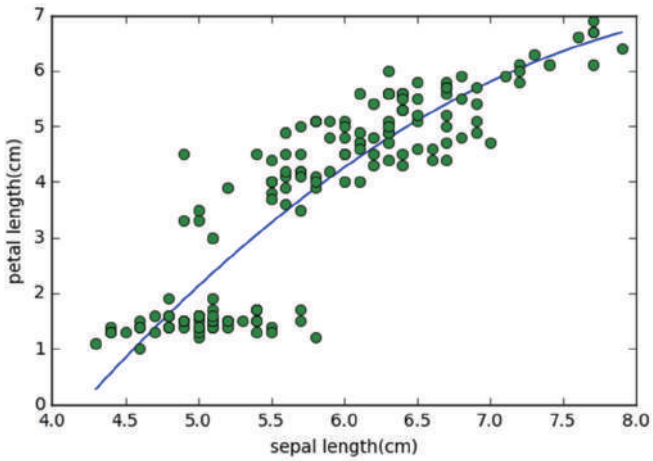


例題：irisデータに対しての最小二乗

- 前回の講義で確認したirisデータの「がく片の長さ」と「花片の長さ」の関係に対して、二次関数（y = ax^2 + bx + c）による非線形回帰を行い、得られた曲線を描きなさい

解答

- 少し上側に膨らむような曲線となっており、データの集合に合わせて描かれていることが確認できる



演習問題

- 相関の演習問題で利用した変数間の関係に関して、回帰分析を行う

- irisのSetosaとVersicolorのがく片の幅と花弁の幅の関係について、がく片の幅を説明変数、花弁の幅を目的変数として、最小二乗法による線形回帰を行え。データ点及び回帰の結果を図示せよ。
- irisのVersicolorとVirginicaのがく片の長さと花弁の幅の関係について、がく片の長さを説明変数、花弁の幅を目的変数として、最小二乗法による二次関数の非線形回帰を行え。データ点及び回帰の結果を図示せよ。

第12回：機械学習全体像

機械学習の全体像を学び
この後の数回に渡る講義のガイドとする

アジェンダ

- 機械学習とは？
- 機械学習で使われる用語の整理
- 機械学習の処理の流れ
  - データの入手
  - データの前処理
  - 機械学習の手法選択
  - パラメータの選択
  - モデルの学習
  - モデルの評価
  - チューニング

機械学習とは？

- 人間が行っていることと同等の学習能力をコンピュータで実現しようとする研究課題の1つ
- より汎用的に言うと、データからルールから見つけ出すアルゴリズムのこと
- 事前に正解データを与えることで学習し、学習した結果から未知のデータに対しても適用できるルールを作り出す手法を教師あり学習という
- 正解データという概念がないデータに対し、データの背後に潜む構造を見つけ出すことでルールを見つける手法を教師なし学習という

機械学習で使われる用語の整理

- 特徴量
- ラベル
- モデル
- 学習
- 学習データ
- 過学習
- チューニング

特徴量

- 特徴ベクトル、などとも呼ばれる
- 物や現象の状態をデータの羅列で表現したもの
- 特徴量自体は何でも良いが、一般的には、機械学習に学習させたいことに対して特徴がよく捉えられているもののの方が良い
- 例えば、人を表現することを考える
  - 表現方法１：身長と体重の2つで表現
    - [170, 50], [175, 70], [165, 70], [180, 80], etc...
  - 表現方法2：50m走のタイム、握力、背筋力
    - [6.5, 50, 100], [7.0, 45, 150], etc...
- 特徴量の数を「データの次元」と呼ぶことも多い

ラベル

- 特徴量で表現されているものが何であるかを表現したもの
  - 主にラベルがついているデータに対して行う学習が教師あり学習と呼ばれる
- ある状態であるか、そうでないか、というラベルを付けることが多い
  - その場合、ラベルが2種類になるため、二値ラベルと呼ばれ、二値のラベルを分類する問題は二値分類問題と呼ばれる
    - 例えば、故障しているか、そうではないか、というラベル
  - 機械学習の分類問題で最も基本的な問題となる
- 複数のラベルをつけても良い
  - その場合は、複数のラベルに分類する多値分類問題と呼ばれる問題となる

学習

- データからモデルを見つけ出す処理の過程のこと
  - 見つけ出す処理の過程にも様々な考え方がり、それが機械学習の各種アルゴリズムとなっている
- 教師あり学習の場合
  - ラベルと特徴量の組み合わせから、あるラベルを表す特徴量にはどのようなルールがあるのかを機械的に見つけることに相当する
- 教師なし学習の場合
  - ラベルに関係なく、与えられた特徴量からデータの背後に潜む構造を探すことに相当する

学習データと過学習

- モデルの構築のために使われるデータのこと
- 機械学習の一般的な流れとしては、以下のとおりになる
  - 手元のデータを、学習データとテストデータ（評価データ）に分ける
  - 学習データにアルゴリズムを適用して学習を行う
  - 学習がうまくいったかの確認にテストデータを使う
- このようにデータを分割する理由として、過学習を割けることが目的となる
- 過学習とは、学習データに対してのみ高い分類精度を示すようなモデルができてしまうこと
  - つまり、学習に使ったデータに対して「のみ」うまく分類ができている状態
  - それを割けるために、評価は学習に使ったデータ以外のものを利用する

チューニング

- モデルの精度を高めるために行う作業
- 機械学習の各種手法には、パラメータが存在する
- パラメータを適切に設定することが機械学習のモデルの精度を高める重要なポイントとなる

機械学習の処理の流れ

1. データの入手
2. データの前処理（加工、整形、尺度の変換など）
3. 機械学習の手法選択
4. パラメータの選択
5. モデルの学習
6. モデルの評価
7. チューニング

機械学習で想定されているデータ

- 教師あり学習の場合は、１つの事例に対してラベルと特徴量がセットになって与えられるデータ
  - [ラベル、(特徴量)]
- 教師なし学習の場合、ラベルは不要。1つの事例に関して特徴量が与えられるデータ
  - [(特徴量)]

前処理

- 手元にあるデータを、機械学習で想定しているデータ形式に変換すること
- 単純にデータ形式を整えるだけでなく、得られている特徴量を加工して新しい特徴量を作ったりもする
  - 例：身長と体重から、BMIの値を作り新しい特徴量にする
- その他、異常値の取扱いを決めたりもする
- 機械学習はデータの整備が一番重要な課題のため、実務的には、前処理に大多数の時間を使う

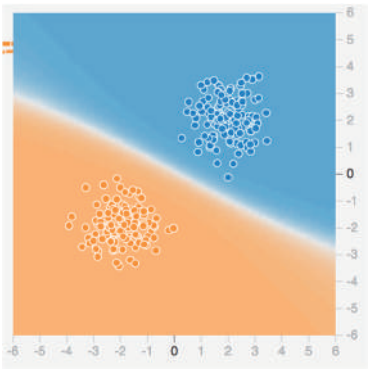
機械学習でできることの整理

- 分類
  - データからラベルを学習し、ラベルを予測する
- 回帰
  - データに当てはまるモデルを検討する
- クラスタリング
  - データの似ているもの同士をまとめて、データの構造を新しく発見するもの
- 次元削減
  - データの次元を落とすことでより本質的な情報を作ったり、可視化しやすくする

「分類」の機械学習手法

- データの集合をラベル毎に分類する線（正確には超平面）を決める

- SVM（Support Vector Machine）
- k-近傍法
- 決定木
- ランダムフォレスト
- etc...

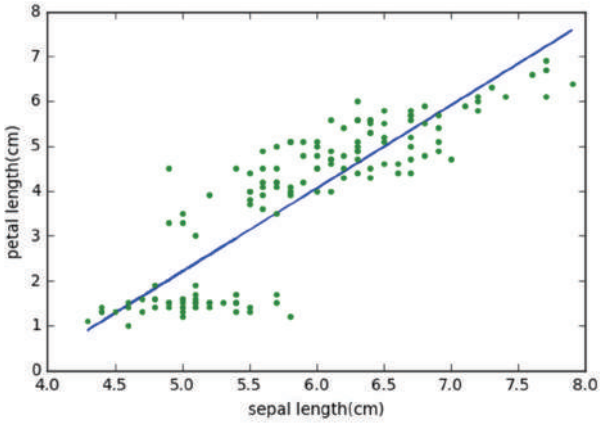


青色と橙色のラベルを分類するような線を引く

「回帰」の手法

- 統計の章で取り上げたもの。機械学習の一手法とみなすことができる
- 回帰によって「予測」ができるようになる

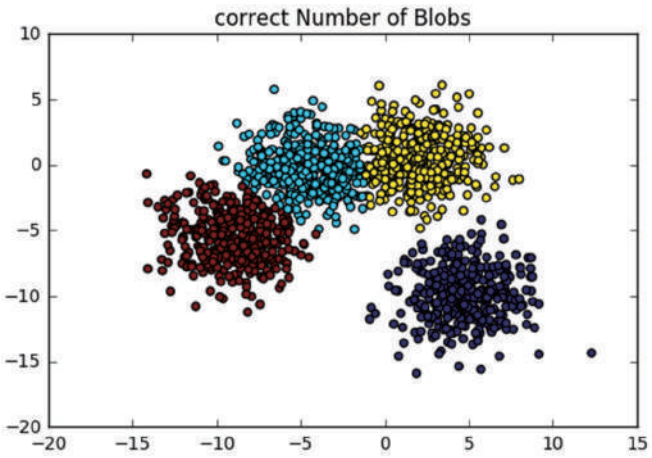
- 線形回帰
- ロジスティック回帰
- リッジ回帰
- etc...



「クラスタリング」の手法

- 教師なしの手法であるため、似通ったもの同士をおおざっぱにまとめるものと考えれば良い

- k-means（k平均法）
- 階層的クラスタリング
- 混合ガウス分布
- etc...



「次元削減」の手法

- 次元削減の主な目的は２つ
  - 特徴量を減らすことでより本質的な情報だけを残す
    - これにより場合によっては、モデルの精度は向上する
  - 可視化可能な次元まで特徴量を減らすことでデータの見える化を行う
    - データを可視化して得られることは多い

- 主成分分析
- 独立成分分析
- 非負値行列因子分解
- etc...

モデルの評価

- 学習がどれくらいうまくいったかを評価するプロセス
- 単純に分類がどれくらいうまくいったかだけに着目していると、過学習に陥ってしまう可能性があるため注意が必要
- 評価するための代表的な指標や考え方は下記の通り
  - Accuracy（精度）
  - Confusion Matrix（混同行列）
  - Precision（適合率）、Recall（再現率）、F値
  - Precision-Recall Curve
  - ROC Curve（Receiver Operating Characteristic Curve：ROC曲線）
- いずれも教師あり学習の手法で適用されるもの
  - 教師なし学習は、そもそも正解のラベルがないため、評価がしにくい

Accuracy（精度）

- 予測されたラベルが、正解ラベルに対してどれくらいあっているかという指標
- 例えば、以下の場合を考える
  - 予測されたラベル：[0, 1, 1, 1, 0]
  - 正解のラベル：[0, 1, 1, 0, 0]
  - 5個中4個正解なので、精度は4 / 5 x 100 = 80% となる
- 精度は一般的な指標であるが、精度のみを見ていると見落とす情報も多い

Confusion Matrix（混同行列）

- 真のラベルと、予測したラベルについて行列で表現したもの
- 精度だけでは見えなかった「正と予測したが実は負だったラベル」「負と予測したが実は正だったラベル」が確認できるようになる
  - 「正と予測したが実は負だったラベル」は偽陽性とも呼ばれ、問題があることを見逃してしまうことに相当するので、問題によっては割けなければいけない間違い
  - 「負と予測したが実は正だったラベル」は偽陰性とも呼ばれ、問題がないものを問題であるとしてしまうことに相当する

混同行列の例

- データ件数5件
- 正しいラベル[0, 1, 0, 1, 0]とする（0は負、1は正のラベルとする）
- 予測したラベル[0, 1, 1, 0, 0]とする

		予測したラベル		
		正	負	合計
正しいラベル	正	1	1	2
	負	1	2	3
	合計	2	3	5

Precision, Recall, F値

- Precision（適合率）とは、正と予測したラベルのうち、実際に正だったものの割合
  - すなわち、正しく予測できたものの、全体に対する割合
- Recall（再現率）とは、実際に正であるラベルのうち、実際に正と予測されたものの割合
  - すなわち、正しく予測されたものの全体に対する割合
- F値とは、適合率と再現率の調和平均
  - 適合率と再現率は一般的にトレードオフの関係にあるため、双方を同時に評価するための指標

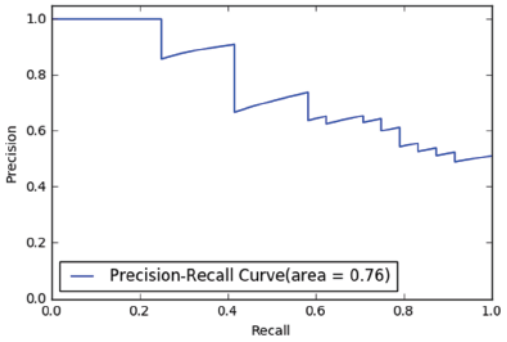
混同行列とPrecision/Recallの関係

		予測したラベル		
		正	負	合計
正しいラベル	正	1	1	2
	負	1	2	3
	合計	2	3	5

混同行列からPrecisionとRecallは算出することができる
・Precision = 1 / 2 = 0.5
・Recall = 1 / 2 = 0.5

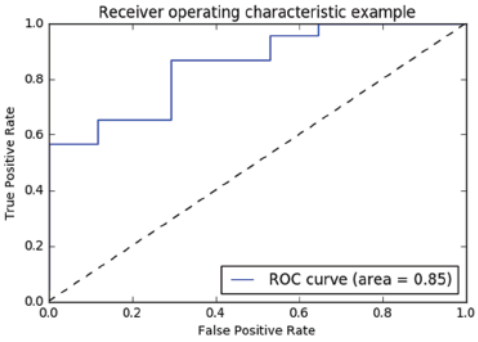
Precision-Recall Curve

- データの件数をランダムに1件から順に増やして行きながら、適合率と再現率を調べてプロットしたもの
- このグラフの下側部分をAUC（Area Under the Curve）と呼び、精度の指標として用いられる



ROC曲線

- 横軸にFalse Positive（偽陽性）の値、縦軸にTrue Positive（正しく分類できた）の値をプロットする
- Precisino-Recall Curveと同じく、データを1件からランダムに取り出し、上記の値を調べ、プロットしていったもの
- 同じく曲線下部の面積はAUCと呼ばれる



AUCについてもう少し

- Precision-Recall Curveでも、ROC曲線でもAUCは0から1までの値を取る
- 完全にきれいな分類ができているものは面積が1となる
- 一方、完全にランダムに分類した際も面積は0.5となる
  - つまり、一般的には0.5より大きな値となる
  - 0.5を下回る場合は、非常に良くない状態

演習問題

- 機械学習を利用している面白いサービスやシステムの例を探してみよ
- そのサービスには、どんなデータを使って、どんな機械学習の手法が使われていると思いますか？特に以下の観点についてまとめてみよ
  - データ
  - 機械学習の手法

第13回：教師あり学習

教師あり学習を中心に機械学習を学ぶ

アジェンダ

- データセットについて
- 一般的なscikit-learnの処理の流れ
- 教師あり学習の代表的な手法
  - k近傍法
  - SVM
- 精度向上のための工夫
  - パラメータチューニング
  - グリッドサーチ
  - 交差検定法
- 結果の評価
- 演習問題

データセットについて

- Scikit-learnには多数のデータがある
- その中の１つにアヤメ（iris）データがある
  - 三種類のアヤメのデータ（つまりラベルが三種類）
    - Setosa, Versicolor, Virginicaの三種類
  - 4つの特徴量
    - がく片の長さ（sepal length）
    - がく片の幅（sepal width）
    - 花弁の長さ（petal length）
    - 花弁の幅（petal width）
- irisデータは最も有名な統計/機械学習用のデータセット
- データセットを使うことで、データ自体の理解は進んでいるため、アルゴリズム実行の理解に集中することができるのが利点である

一般的なScikit-learnの流れ

1. データセットを用意する
2. アルゴリズムのクラスをインスタンス化
   - インスタンス化のタイミングで各種パラメータを渡す
3. アルゴリズムのクラスには、学習のための関数fit()が用意されているため、データを与えて学習させる
4. アルゴリズムのクラスには、予測のための関数predict()が用意されているため、データを与えて分類させる
5. 分類結果を評価する
6. もっと良いパラメータやアルゴリズムがないかを探す

※教師あり学習の場合。教師なし学習や次元削減などの場合はやや異なる

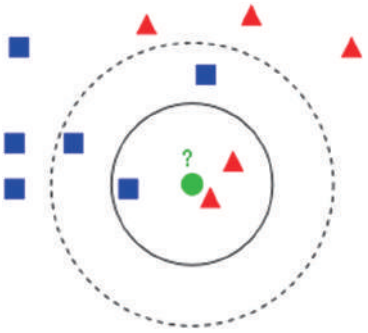
教師あり学習の代表的手法

- 分類
  - データからラベルを学習し、ラベルを予測する
  - k近傍法
  - SVM
  - ランダムフォレスト
  - etc...
- 回帰
  - データに当てはまるモデルを検討する
  - 線形回帰
  - ロジスティック回帰
  - etc...

- 本講義では主に分類の手法について取り扱う

k近傍法

- k近傍法は距離の近いk番目までの点から、どのクラスに分類されるかを判断する手法
- シンプルながら強力な手法である一方、計算量は多くなる

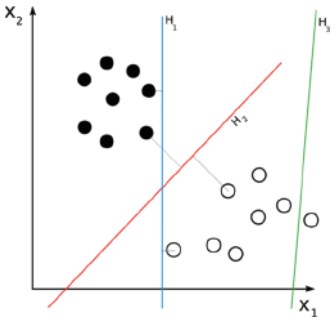


・緑のデータを分類したい
・近傍数kを3とする
・赤い三角の方が数が多いため、緑のデータは赤い三角側のクラスと分類される
・例えば近傍数kを5とすると結果は変わる

出典：https://upload.wikimedia.org/wikipedia/commons/thumb/e/e7/KnnClassification.svg/220px-KnnClassification.svg.png

SVM（サポートベクターマシン）

- クラス間の距離（マージン）を最大化するような超平面を学習する手法
- カーネル法という手法を使うことで、非常に高い分類精度を達成する手法の１つ
  - 理論面は非常に複雑だが、イメージは比較的簡単



【直感的説明】
・2つのグループを分けたい
・最も良いのは青色、赤色、緑色のどの直線か？
・２つのクラスが最も離れるように引ける直線が「良い」とする
・その場合、赤色の直線が最も良い

出典：https://upload.wikimedia.org/wikipedia/commons/2/20/Svm_separating_hyperplanes.png

精度向上のための工夫

- 精度向上のために以下のような工夫を行う
- パラメータチューニング
  - アルゴリズムによってパラメータの種類や数は異なる
  - k近傍法の場合は、近傍の数
  - SVMの場合は、カーネルなど多数ある
- パラメータチューニングの一方法としてグリッドサーチと呼ばれる手法がある

パラメータチューニング

- SVMの場合
  - パラメータはドキュメントを参照する
  - http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
- パラメータを変化させると、結果が変わることを確認しよう

- グリッドサーチは考えられるパラメータの組み合わせを全て試す方法
  - 探索したいパラメータ及び、その数値の候補を与える
  - 力技で全てのパターンを試し「最も良い」パラメータを選択する

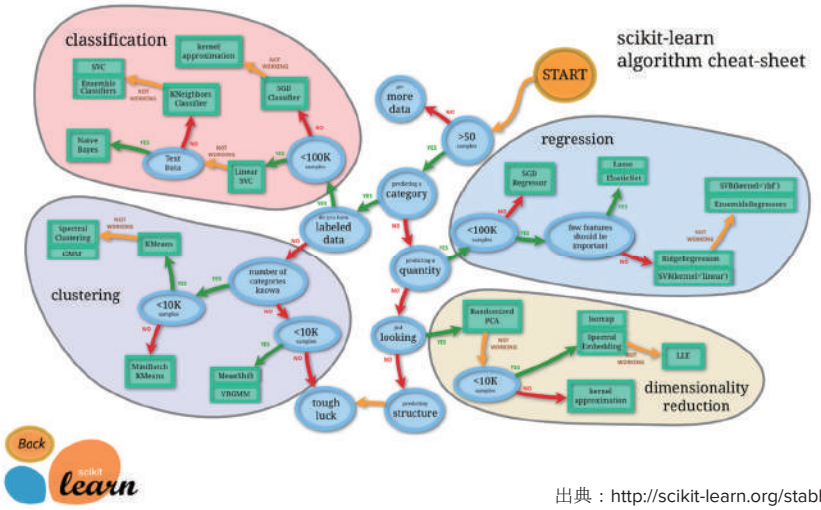
- 最も良いパラメータとは何か、を決めるための指針が必要

学習結果の評価

- 今学習されたモデルがどれくらい良いのかについて評価する
- パラメータチューニングは指針がないと実行できない
  - パラメータチューニングの指針に相当するもの
- いくつかの精度評価の方法がある
  - Accuracy
  - Confusion Matrix
  - Precision Recall Curve
  - ROC Curve

チートシート

- どのような時にどういった手法を使うかの判断材料となる



出典：http://scikit-learn.org/stable/tutorial/machine_learning_map/

演習課題

- scikit-learnに付属しているデータセットの１つである手書き認識文字データのdigitを使って、精度高く手書き文字を認識させるモデルを作りなさい
  - digitは8x8=64ピクセルで表現された手書き文字
  - 0から9までの10クラス
- 前項で紹介したチートシートも参考にすること

第14回：教師なし学習

教師なし学習を中心に機械学習を学ぶ

アジェンダ

- 教師なし学習とは？
- 教師なし学習の代表的な手法について
  - クラスタリング
    - k-means
  - 次元削減
    - 主成分分析
- 演習問題

機械学習とは？

第12回より再掲

- 人間が行っていることと同等の学習能力をコンピュータで実現しようとする研究課題の1つ
- より汎用的に言うと、データからルールから見つけ出すアルゴリズムのこと
- 事前に正解データを与えることで学習し、学習した結果から未知のデータに対しても適用できるルールを作り出す手法を教師あり学習という
- 正解データという概念がないデータに対し、データの背後に潜む構造を見つけ出すことでルールを見つける手法を教師なし学習という

教師なし学習について

- 教師なし学習はラベルを持たないデータに対する機械学習の手法を指す
- 明確に正解と言われるものがないため、結果に対して何らかの指標を元に評価することはできない
- データの背後にある構造を探し、それを元にデータをまとめたり（クラスタリング）、次元を減らす（次元削減）を行うような手法群

クラスタリング

- 与えられたデータを何らかのルールに従って分類する手法
  - ラベルなどの外部情報なく、データそのものから特徴を見つける方法

- 大きく分類すると階層的クラスタリングと非階層的クラスタリングがある
  - 階層的クラスタリングの代表的手法はウォード法
  - 非階層的クラスタリングの代表的手法はk-means

- 本講義ではクラスタリングの代表的な手法として、k-meansを紹介する

k-means

一般的に下記のようなアルゴリズムである

1. 各データをランダムにクラスタ（クラス）に割り振る
2. 割り振ったデータからクラスタの中心を求める
3. 中心と各データ点の距離を求め、最も近いクラスタの中心に割当て直す
4. 2,3を繰り返し、一定の変化量以下となったら処理を打ち切る

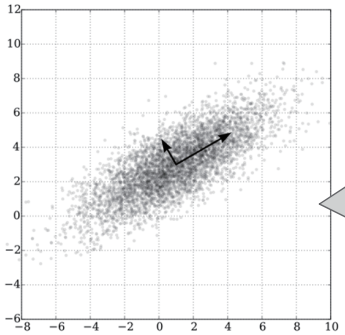
非常にシンプルなアルゴリズムだが、同じクラスたが近い距離にあるような場合には非常にうまく働く

次元削減

- 特徴量が多くある（高次元）データを、全体の情報を損なわずに低次元のデータに変換すること
  - 1000ある特徴量を3つの特徴量に絞り込む
    - 単純に1000の中から3つを選ぶ場合、特徴選択(Feature Selection)と呼ぶ
    - 1000の特徴量を合成し、3つの特徴量を新たに作る場合、次元削減(Dimensionaly Reduction)と呼ぶ
- メリット
  - 不要な情報を減らすことができるため、本質的な情報のみが残される
  - 低次元になることにより、可視化を行うことができる
- デメリット
  - 次元を落としすぎると必要な情報まで削られてしまい、全体の情報量が減ってしまう
  - 適切な次元の落とし方は簡単に求まらない

主成分分析

- 次元削減の手法で最も古典的であり有名な手法
- 特徴量の中で、相関の少ないものを選び出し、新たな特徴量を作ることにより、少ない特徴量でデータ全体の特徴を表現させる方法

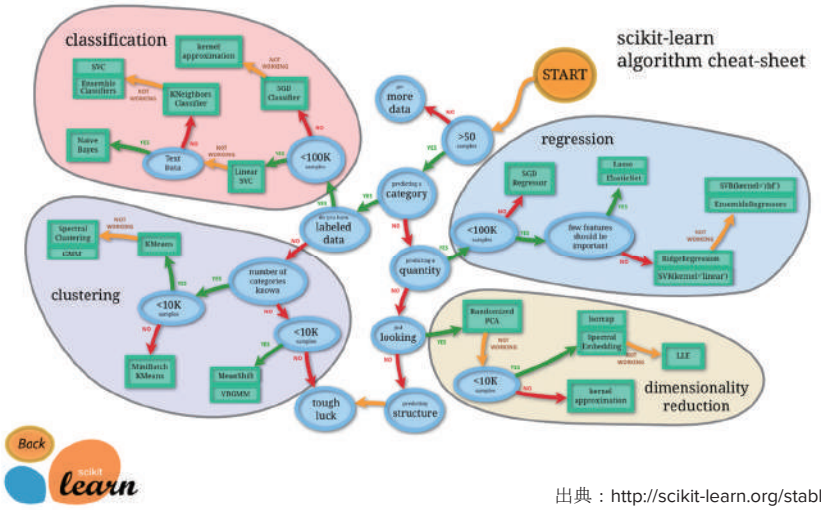


【直感的説明】
・散らばってるデータから見て最も説明力の高い軸を新たに探す
・この例の場合、楕円状に散らばっているデータの真ん中を通る2軸
・数学的には共分散行列の固有値と固有ベクトルを計算することで求めることができる

参考：https://commons.wikimedia.org/wiki/File:GaussianScatterPCA.svg#/media/File:GaussianScatterPCA.svg

チートシート

- どのような時にどういった手法を使うかの判断材料となる



出典：http://scikit-learn.org/stable/tutorial/machine_learning_map/

演習問題

- irisデータのがく片の長さと花片の長さをデータとしてk-meansを実行せよ
  - 本来のクラスタリングの用途とは異なるが、4種類のアヤメデータがあることが分かっているのでkは4とする
  - 結果は描画してみること
- irisデータのがく片の幅と花片の幅をデータとしてk-meansを実行せよ
  - 同じくkは4とし、結果を描画してみること

第15回：大規模データ処理

大規模データ処理に関して学ぶ

アジェンダ

- 大規模データとは？
- Hadoopの基礎知識
- SparkとHadoop
- Spark自体の歴史
- Sparkの基本アイデアと実行環境
- 演習問題

大規模データとは？

- ものすごくたくさんのデータ
  - 大きすぎて1台のマシンで処理しようとしても終わらないデータ量のこと
- 当然、1台のマシンで処理ができないことが多い
  - もしくは処理するのにとんでもなく長い時間がかかる

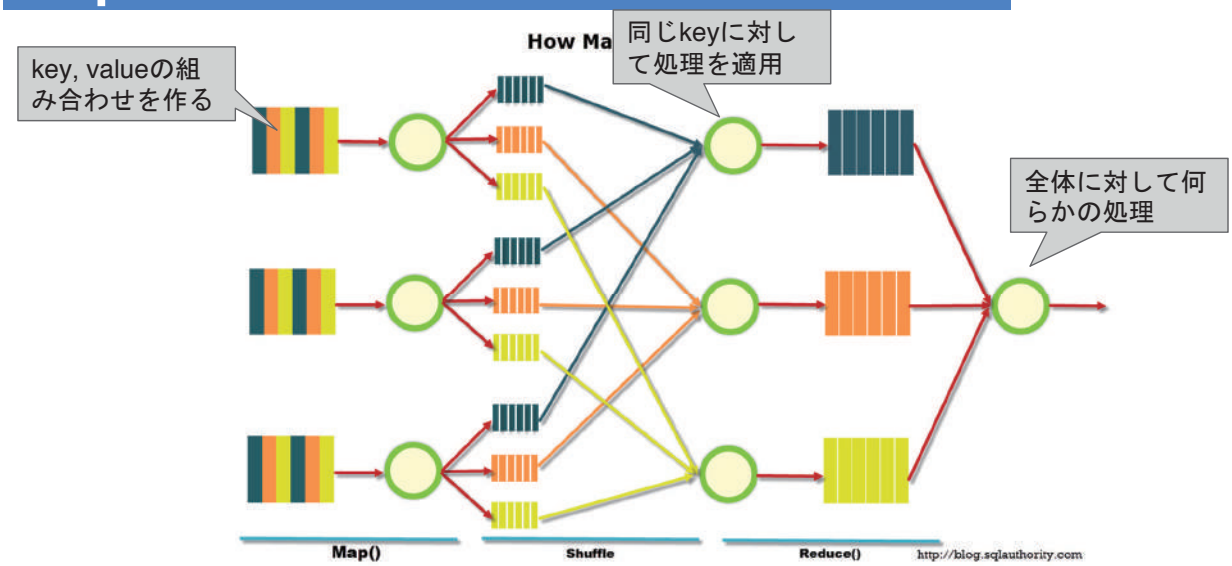
- 大規模データを実務で使えるように処理するには、これまでとは異なった分散処理などに取り組む必要がある
  - 分散処理とは、複数台のマシンで1つの処理を分担して行うこと
  - 複数のマシンで処理を行うため、1台のマシンで行うより処理が早く終る

Hadoop

- 大規模データ処理のための分散処理基盤として最も有名なものとしてHadoopがある
- Googleが出した2つの論文が元になり、オープンソース化された
  - 分散処理のプログラミングパラダイムである「MapReduce」
  - 分散ファイルシステムである「Google File System(GFS)」

- Hadoopは上記2つの仕組みも取り入れて、オープンソース化されたもの
  - MapReduce
  - HDFS（Hadoop Distributed File System）

MapReduceのイメージ



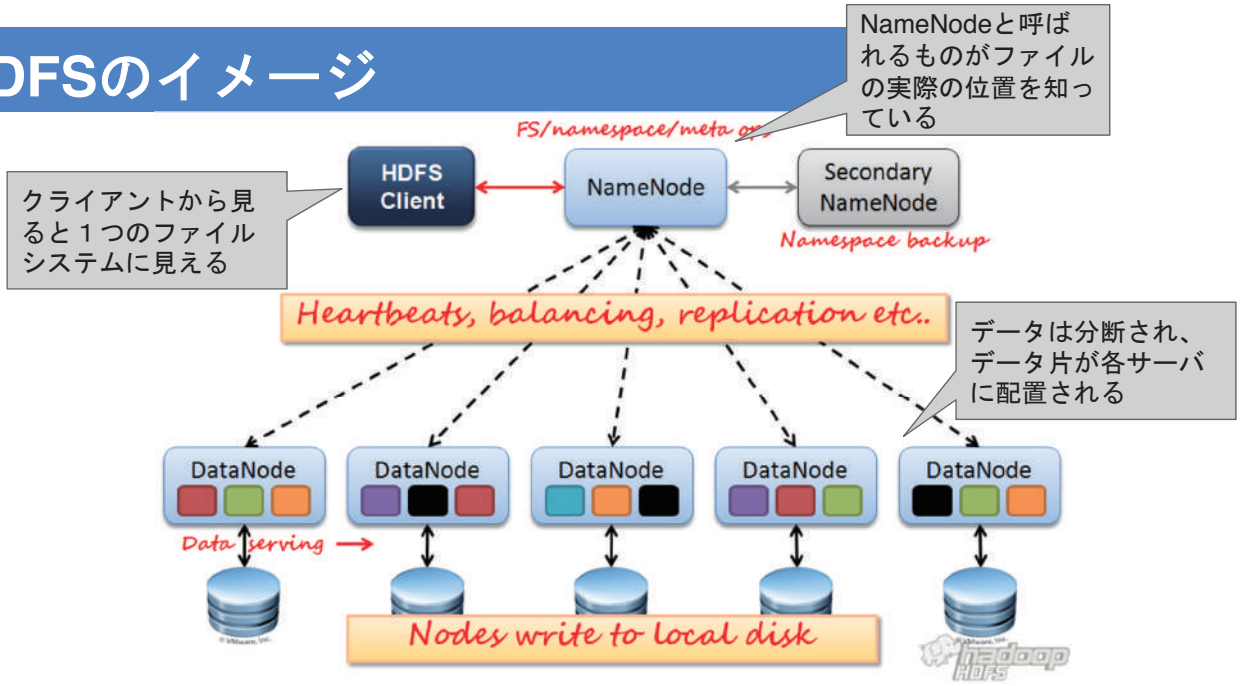
出典：http://blog.sqlauthority.com/2013/10/09/big-data-buzz-words-what-is-mapreduce-day-7-of-21/

MapReduceについて

- 大規模データ処理に強みを持つ「プログラミングモデル」
  - ある程度以上のサイズのデータの塊がたくさんある状態に強い
  - データは最初から最後までアクセスする処理
    - シーケンシャルアクセス
  - サーバ台数が増えれば増えるほどに、基本的には線形に処理能力がます

- 一方、下記のような処理は苦手
  - 小さなデータの塊がたくさんある状態
  - データの中の特定のデータをピックアップする処理
    - ランダムアクセス

HDFSのイメージ



出典：https://yoyoclouds.wordpress.com/tag/hdfs/

HDFSについて

- 「分散ファイルシステム」の一種
  - サーバ群を1つのファイルシステムのように扱える
- HDFSの特徴
  - データをある程度の塊に分けてサーバに分散して配置
  - あるサーバが壊れてもデータ自体が壊れないように、いくつかのサーバに複製データを配置
  - サーバを足すと保持できるデータ量が線形に増える

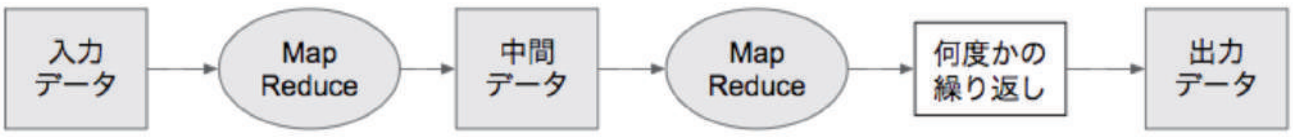
- マシン間でデータのコピーなどが発生するために、データの読み書きそのものは速くない

SparkとHadoop（1/2）

- 当初は、大規模データ処理については実用上Hadoopで十分だった
- Hadoopによって大規模データの活用が進んだ結果下記のようなニーズが出てきた
  - より短い時間で大規模データ処理を行いたい
  - 大規模データ処理に機械学習などのより複雑な処理を行いたい
- これらのニーズは、MapReduceが苦手とする処理であった
- そのニーズを満たすためにいくつかのプログラミングパラダイムが出てきたが、その中で現在最も活用が進んでいるのがSpark

SparkとHadoop（2/2）

- 機械学習などのデータ処理では多数の繰り返し処理が走る
  - MapReduceは、一度のMapReduce処理が終わるたびに、結果を一度中間データとしてディスクに書き出す必要があるため、繰り返し処理がそもそも苦手
- データ分析はトライアンドエラーで様々な処理を行うアドホック処理が業務の多数を攻める
  - MapReduceは対象のデータ全てを処理することに特化しており、データの一部に対して様々な処理を掛けていくことは苦手
- その結果、データ分析により特化した処理系としてSparkが誕生した



MapReduceの処理のイメージ

Sparkの歴史

- 2009年にカルフォルニア大学バークレイ校のAMPLabの1プロジェクトとして誕生
- 当初はBDAS（the Burkeley Data Analytics Stack）の1プロジェクトとして取り組まれていた
- 2010年頃にオープンソース化された
- 2013年にはASF（Apache Software Foundation）に寄贈され「Apache Spark」となる
- 2016年7月頃にVer2.0が出た

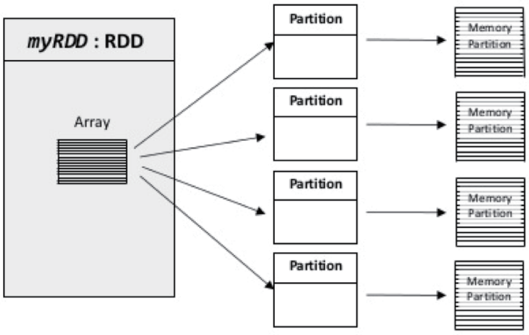
Sparkの基本アイデア

- RDD(Resilient Distributed Dataset)と呼ばれるデータ構造
- DAG(Directed Acyclic Graphic)と呼ばれる有向非循環グラフを利用した計算モデル

- その他にも色々あるが、大きなものは上記２つ
  - データが実際に必要になるまで計算を実行しない遅延評価
  - 途中の計算処理を保持することで、再計算のコストを下げるキャッシュ
  - etc...

RDD

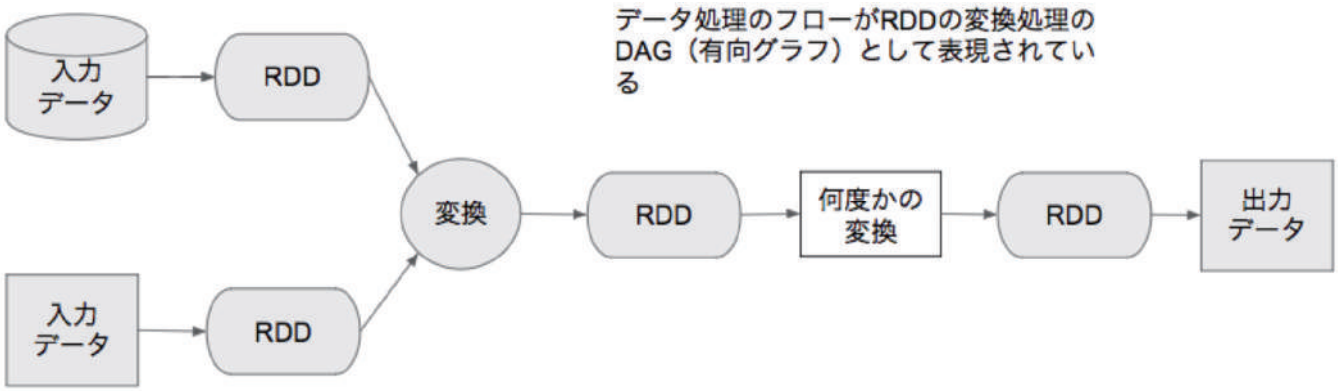
- immutableな分散コレクション
  - 一度作成されると変更されないことが保証されているデータ
  - Partitionという単位で分割され、各サーバに分散して配置される



出典：http://www.slideshare.net/sparkInstructor/apache-

DAGによる計算モデル

- Sparkの計算モデルは、DAGと呼ばれる（分岐はあるが）一方通行のグラフを利用したRDDの変換の繰り返しである



Sparkの実行環境

- Spark自体はScalaで実装されている
- 実行自体はJVMがあれば動作する
- 複数の言語のAPIが提供されているため、アプリケーション開発には下記の言語が使える
  - Python
  - Java
  - Scala
  - R

Sparkのコンポーネント

- 様々なデータ分析タスクにとって使いやすいコンポーネントも提供されている
- SparkSQL
  - 構造化データを扱うことに特化したコンポーネント
  - 多くの人が使い慣れているSQLを利用して大規模データにアクセスできる
- GraphX
  - グラフ構造を処理にするのに便利な機能を提供している
- SparkStreaming
  - ストリーミング処理に特化した機能を提供している
- MLlib
  - 機械学習のための各種機能を提供している

Sparkの利用者とそのメリット

- データサイエンティスト
  - 慣れ親しんだPythonやRを利用して、大規模データを扱えるため、非常に作業効率が良い
- データエンジニア
  - データの出し入れや前処理などのタスクが全て一元化されて非常に効率的に作業が進められる
- アプリケーションエンジニア
  - 複数の言語に対してのAPIが提供されているため、自身が得意としている言語で開発できる可能性が高い

※データ分析に係る多くの人にとって有益なものとなっている

演習問題

- 大規模データと言えるデータの例を考えて自由にまとめよ
- まとめる際には下記のような観点を加えよ
  - どの領域の
  - どのようなデータが
  - なぜ大規模データとなるのか

- 余力がある人は実際にSparkをインストールして動かしてみよ
  - 分散処理のためのミドルウェアだがスタンドアロンモードがあり、一台のマシン上でも動作させることができる

平成 30 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

平成 30 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

機械学習Ⅰ

平成 31 年 3 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