必要準備

WiFi 接続

SSID: fr-other
PASS: vWmQBs3ETAU%Kdk8G

Aidemy Meetup Vol.2

2017/7/15

本日の目標

- 機械学習(Deep Learning)で今何ができるのかを知る
- 実際にプログラムを動かして，機械学習を体感する

タイムテーブル

- 18:15~ 本イベントの概要・自己紹介
- 18:30~ ディープラーニングでできること
- 18:40~ Hands On
- 19:30~ Hands On解説
- 20:00~ フリーワーク
- 21:30~ 本イベントのおさらい

必要なもの

- PC
- Wi-Fi環境
- AWSアカウント(なければ今日作っていただきます)
- 質問受け付け・資料共有

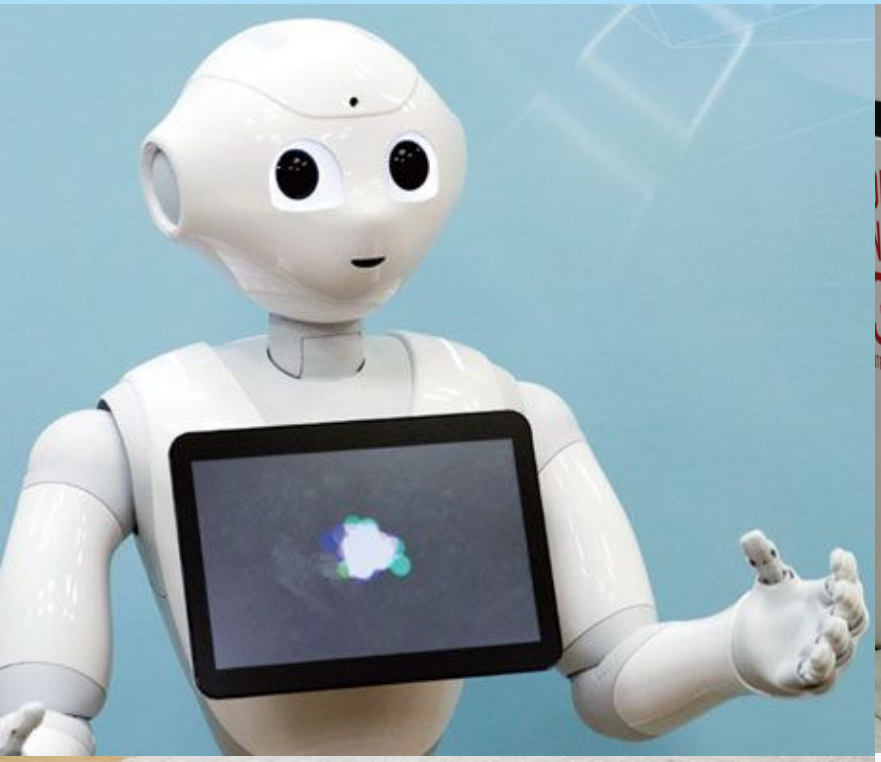
自己紹介

- お名前，ご所属
- AIでどんなことをしたいか

イントロダクション

AI・機械学習・Deep Learning

AI・機械学習・ディープラーニング



人工知能(AI)

=

人間の知能をコンピュータで模倣

学習・理解・推論・計画…

機械学習

=

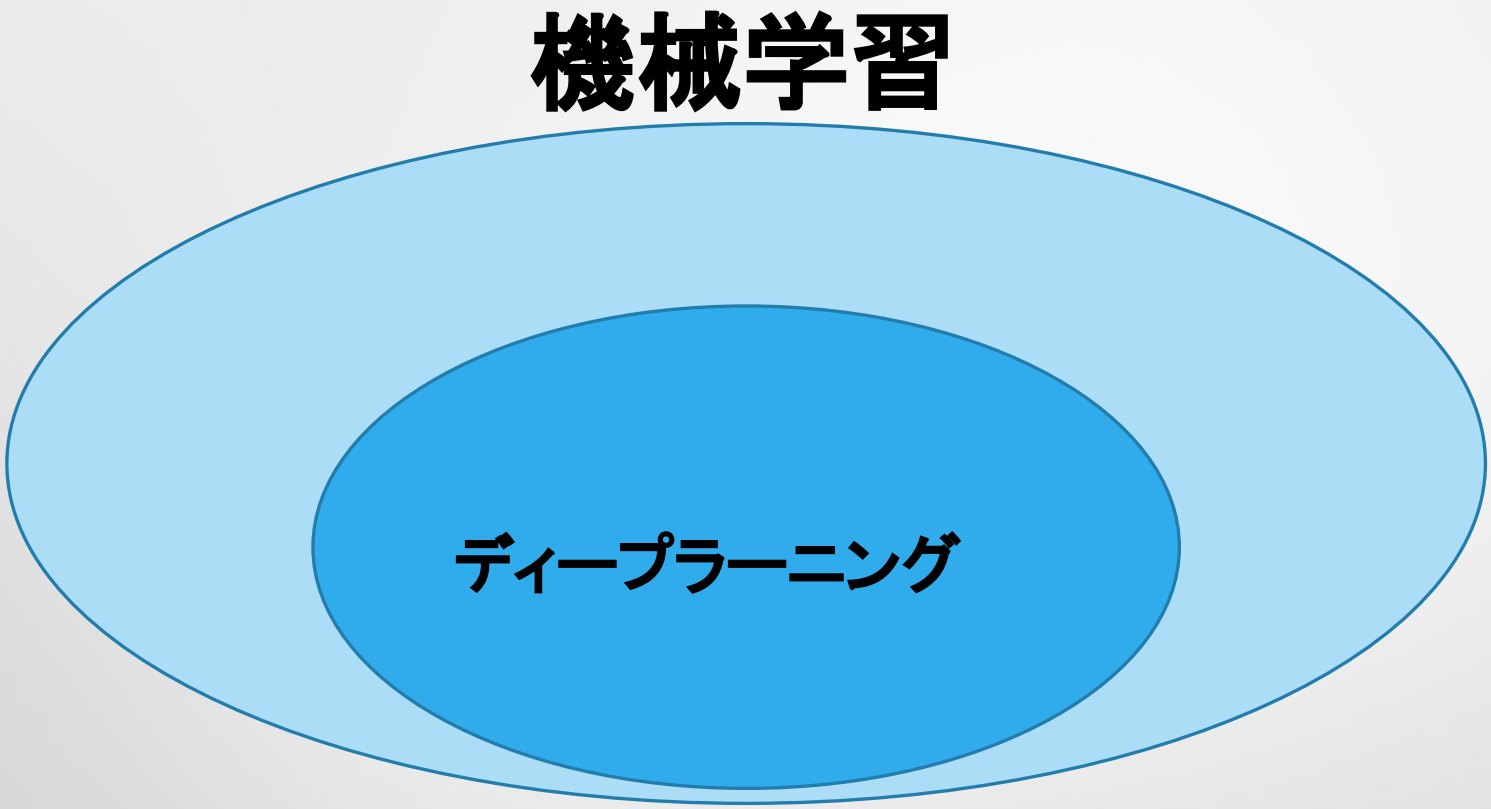
データから知識やルールを

自動的に獲得し

未知のデータを予測・分類する

手法・学問体系

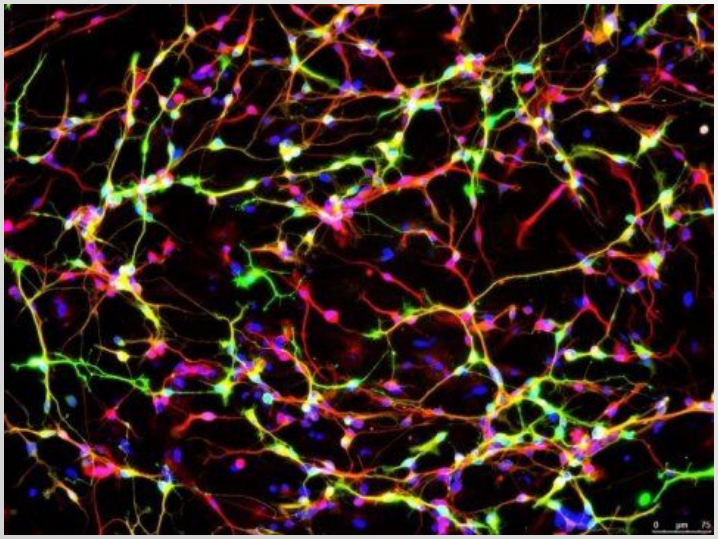
AI・機械学習・ディープラーニング



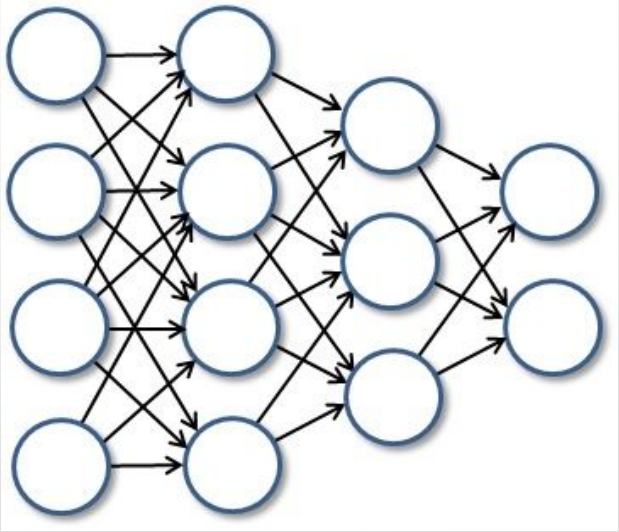
ディープラーニング = 機械学習手法の一つ

ディープラーニングとは

- 汎用人工知能(AI)に今一番近い手法
- 人間の脳を模した数理モデル
- 同じ仕組みで，様々な用途



ニューロン
(シナプス)



ニューラルネット模式図

機械学習で今できること[1]

- スパム検知
- 数字認識
- 会話理解
- 顔検出
- レコメンド機能
- 医療診断

etc ...

[1] http://machinelearningmastery.com/practical-machine-learning-problems/

機械学習やってみよう！

教師データあり 回帰 (一般化)線形回帰 ロジスティック回帰 サポートベクターマシーン(SVM) 木 決定木（CART) 回帰木 ランダムフォレスト 勾配ブースティング木 ニューラルネットワーク(NN) パーセプトロン 畳み込みニューラルネットワーク(CNN) 再起型ニューラルネットワーク(RNN) 残差ネットワーク(ResNet) ベイズ 単純ベイズ（ナイーブベイズ) 時系列 AR,MA,(S)ARIMAモデル状態空間モデル クラスタリング k近傍法(KNN) アンサンブル学習 ブースティング バギング 教師データなし クラスタリング 階層型クラスタリング(ユークリッド距離＊ウォード法など) 非階層型クラスタリング（k-meansなど) トピックモデル（LDAなど) NN 自己組織化マップ（SOM) その他 アソシエーション分析 協調フィルタリング（アイテムベース、ユーザベースなど)カーネル密度推定 scikit-learn Python Jupyter R CPU/GPU 並列化 Hadoop Spark Chainer TensorFlow Theano Keras libsvm 時系列解析 Jubatus EMアルゴリズム 確率的勾配降下法 Karush-Kuhn-Tucker condtion マージン最大化 ソフトマージン カーネルトリック 非線形写像 超平面 クロスエントロピー イェンゼンの不等式 最尤推定 MAP推定 ラプラススムージング NP困難 対数尤度 最尤推定 MCMC法 ギブスサンプリング Lasso Ridge スパース回帰 ロバスト回帰 自己組織化 バンディットアルゴリズム

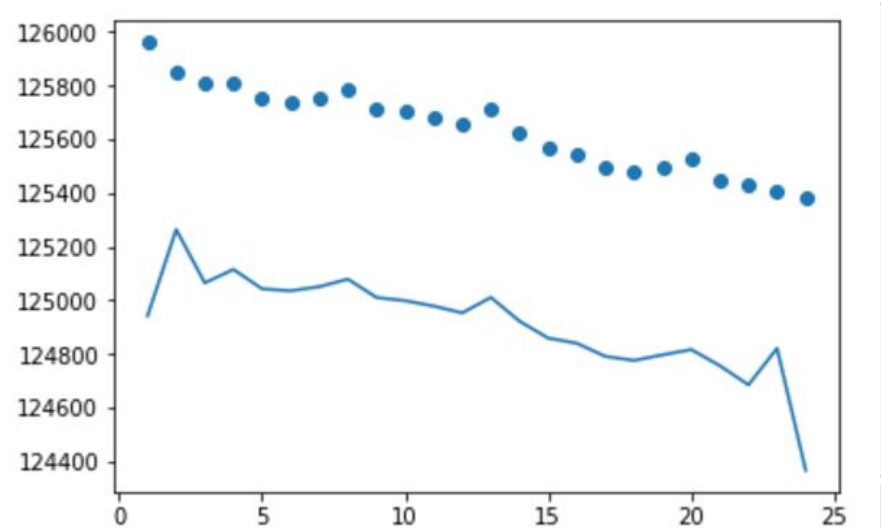
機械学習やってみよう！

教師データあり 回帰 (一般化)線形回帰 ロジスティック回帰 サポートベクターマシーン(SVM) 木 決定木（CART) 回帰木 ランダムフォレスト 勾配ブースティング木 ニューラルネットワーク(NN) パーセプトロン 畳み込みニューラルネットワーク(CNN) 再起型ニューラルネットワーク(RNN) 残差ネットワーク(ResNet) ベイズ 単純ベイズ（ナイーブベイズ) 時系列 AR,MA,(S)ARIMAモデル状態空間モデル クラスタリング k近傍法(KNN) アンサンブル学習 ブースティング バギング 教師データなし[illegible]リング 階層型クラスタリング(ユークリッド距離＊ウォード法など) 非階[illegible]スタリング（k-meansなど) トピックモデル（LDAなど) NN 自己組織化[illegible]OM) その他 アソシエーション分析 協調フィルタリング（アイテムベ[illegible]ーザベースなど)カーネル密度推定 scikit-learn Python Jupyter R CP[illegible] 並列化 Hadoop Spark Chainer TensorFlow Theano Keras libsvm[illegible]列解析 Jubatus EMアルゴリズム 確率的勾配降下法 Karush-Kuhn-[illegible] condtion マージン最大化 ソフトマージン カーネルトリック 非線形写像 超平面 クロスエントロピー イェンゼンの不等式 最尤推定 MAP推定 ラプラススムージング NP困難 対数尤度 最尤推定 MCMC法 ギブスサンプリング Lasso Ridge スパース回帰 ロバスト回帰 自己組織化 バンディットアルゴリズム

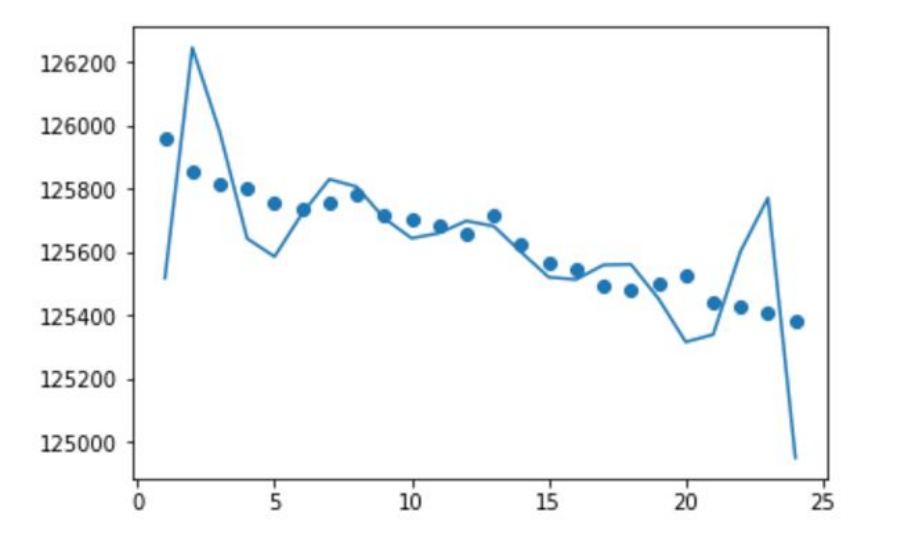
機械学習やってみよう！

- とりあえず使ってみよう. . . !

```
from sklearn.kernel_ridge import KernelRidge

x = np.array(data1314.index).reshape(-1,1)
y = data1314.values

clf = KernelRidge(alpha=0.01, kernel='rbf')
clf.fit(x, y)

p = clf.predict(x)

plt.scatter(x, y)
plt.plot(x, p)

print(clf.score(x, y))
```



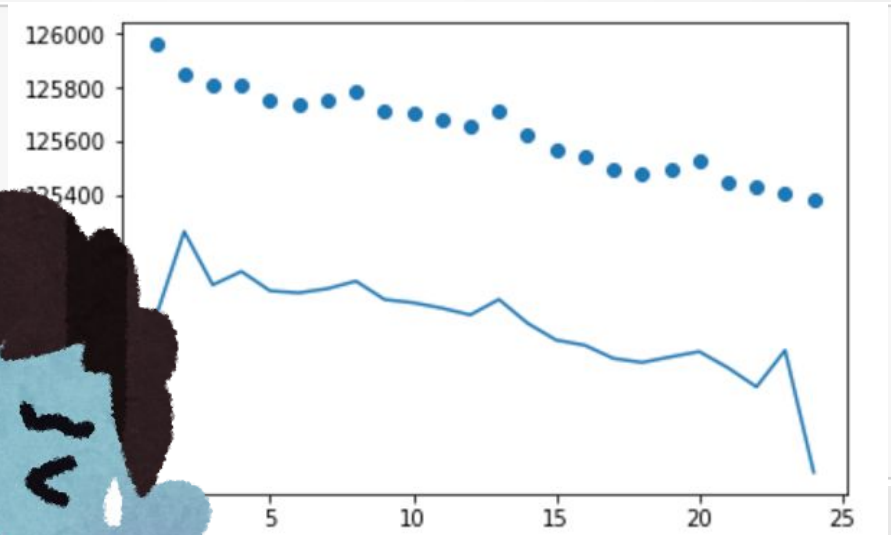

```
from sklearn.kernel_ridge import KernelRidge

x = np.array(data1314.index).reshape(-1,1)
y = data1314.values

clf = KernelRidge(alpha=0.001, kernel='rbf', gamma=0.03)
clf.fit(x, y)

p = clf.predict(x)

plt.scatter(x, y)
plt.plot(x, p)

print(clf.score(x, y))
```



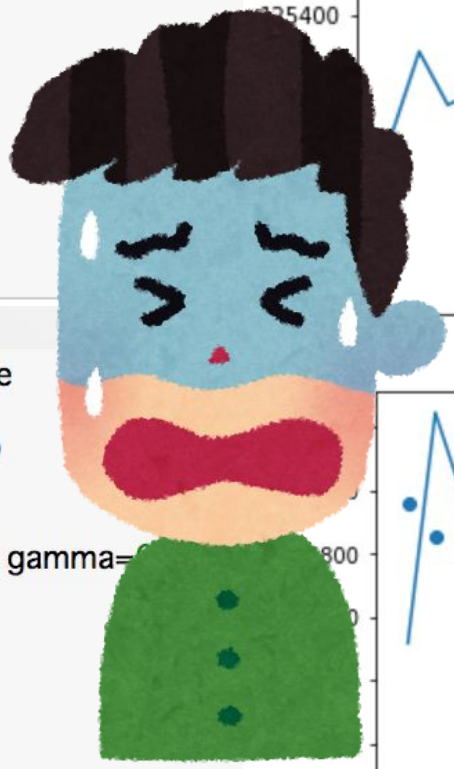
機械学習やってみよう！

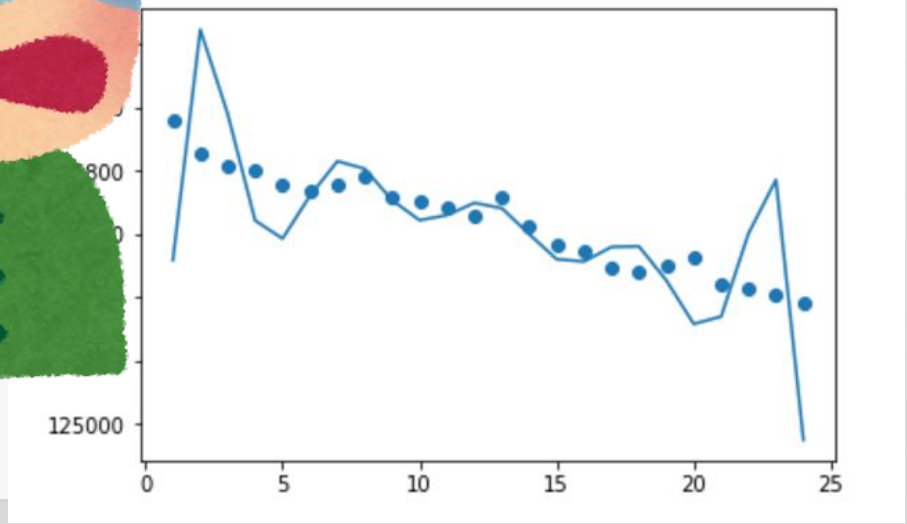
- とりあえず使ってみよう. . . !

```
from sklearn.kernel_ridge import KernelRidge

x = np.array(data1314.index).reshape(-1,1)
y = data1314.values

clf = KernelRidge(alpha=0.01, kernel='rbf')
clf.fit(x, y)

p = clf.predict(x)

plt.scatter(x, y)
plt.plot(x, p)

print(clf.score(x, y))
```




```
from sklearn.kernel_ridge import KernelRidge

x = np.array(data1314.index).reshape(-1,1)
y = data1314.values

clf = KernelRidge(alpha=0.001, kernel='rbf', gamma=
clf.fit(x, y)

p = clf.predict(x)

plt.scatter(x, y)
plt.plot(x, p)

print(clf.score(x, y))
```



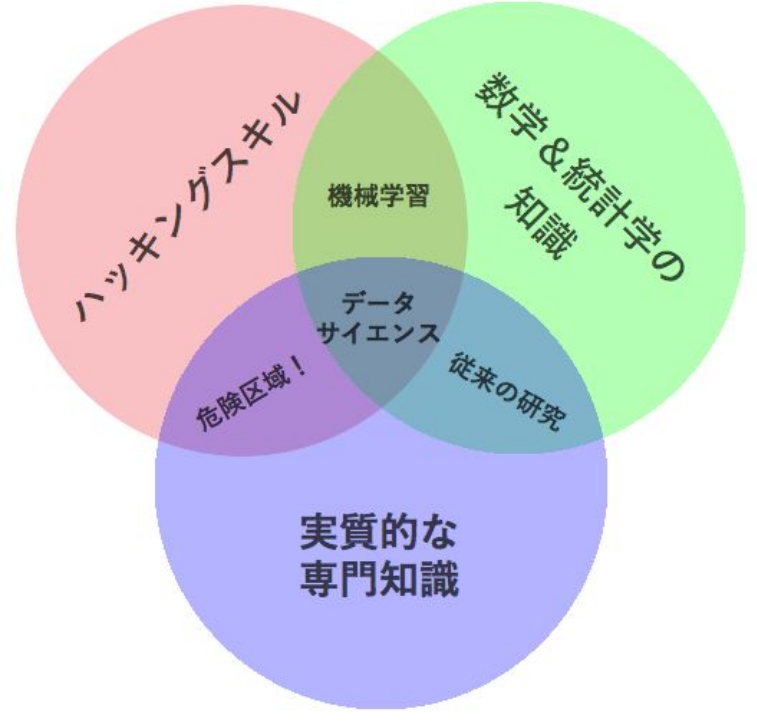


機械学習の壁

- 手法が多すぎてわからない
- 理論が難しくてわからない
- フレームワーク，APIは充実しているが
  うまく使いこなせない

機械学習の壁

- 分野横断的
  - 数学+プログラミング(+専門知識)



[1] http://drewconway.com/zia/2013/3/26/the-data-science-venn-diagram

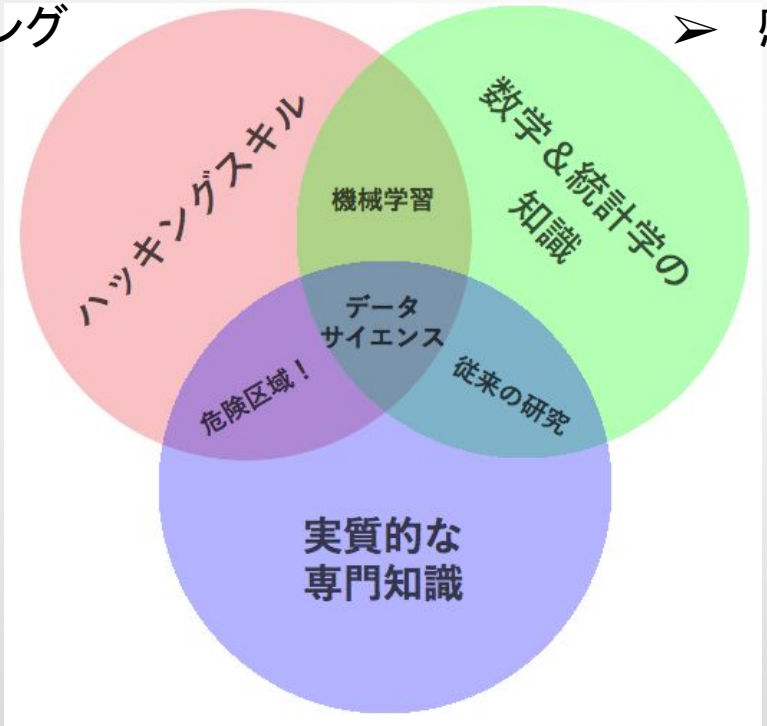
実際に必要なこと

API/フレームワーク

- 直観的なプログラミング

直観的理解

- 感覚的な理論解釈



ドメインナレッジ

- 扱うデータに対する知識

[1] http://drewconway.com/zia/2013/3/26/the-data-science-venn-diagram

実際に必要なこと

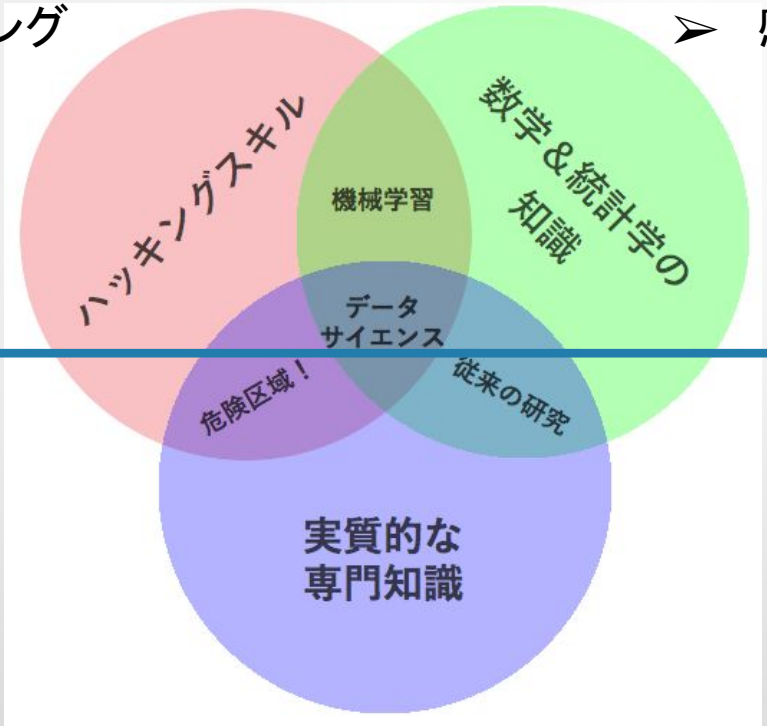
今回の目標

API/フレームワーク

- 直観的なプログラミング

直観的理解

- 感覚的な理論解釈



ドメインナレッジ

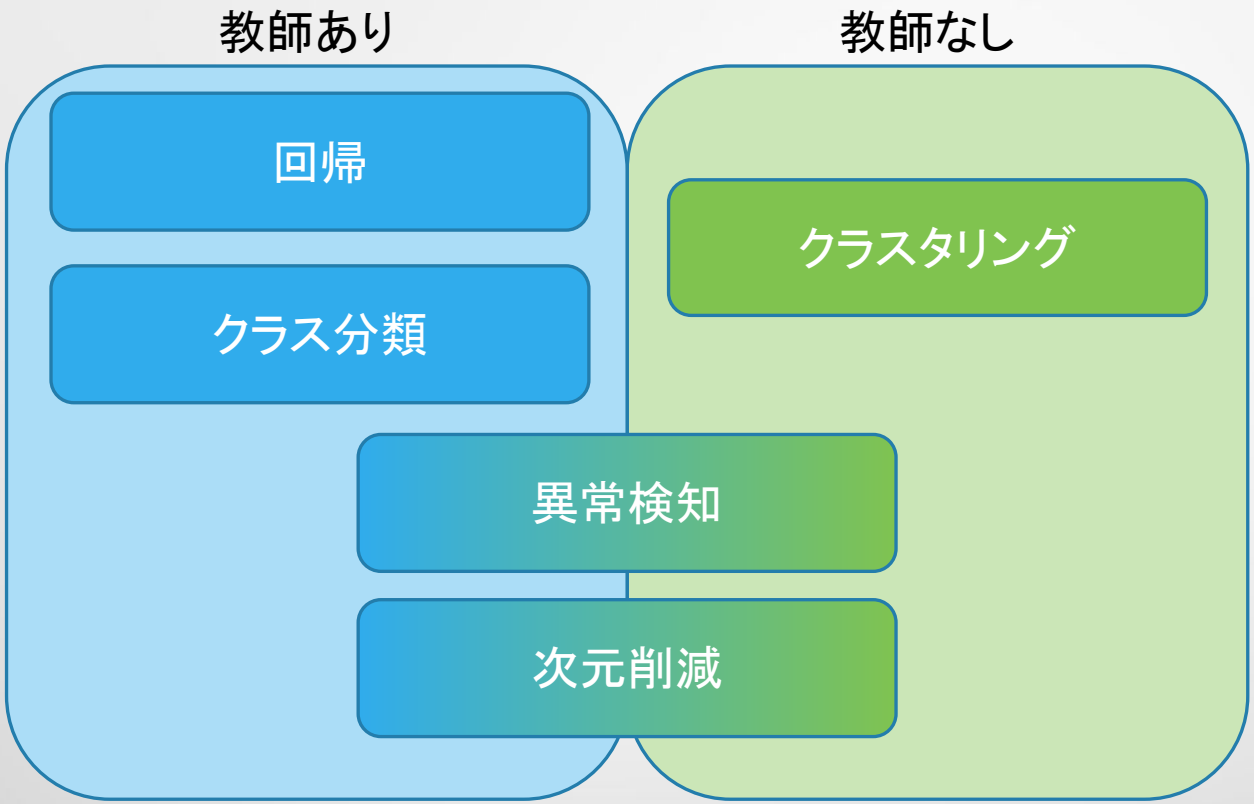
- 扱うデータに対する知識

[1] http://drewconway.com/zia/2013/3/26/the-data-science-venn-diagram

本日の課題

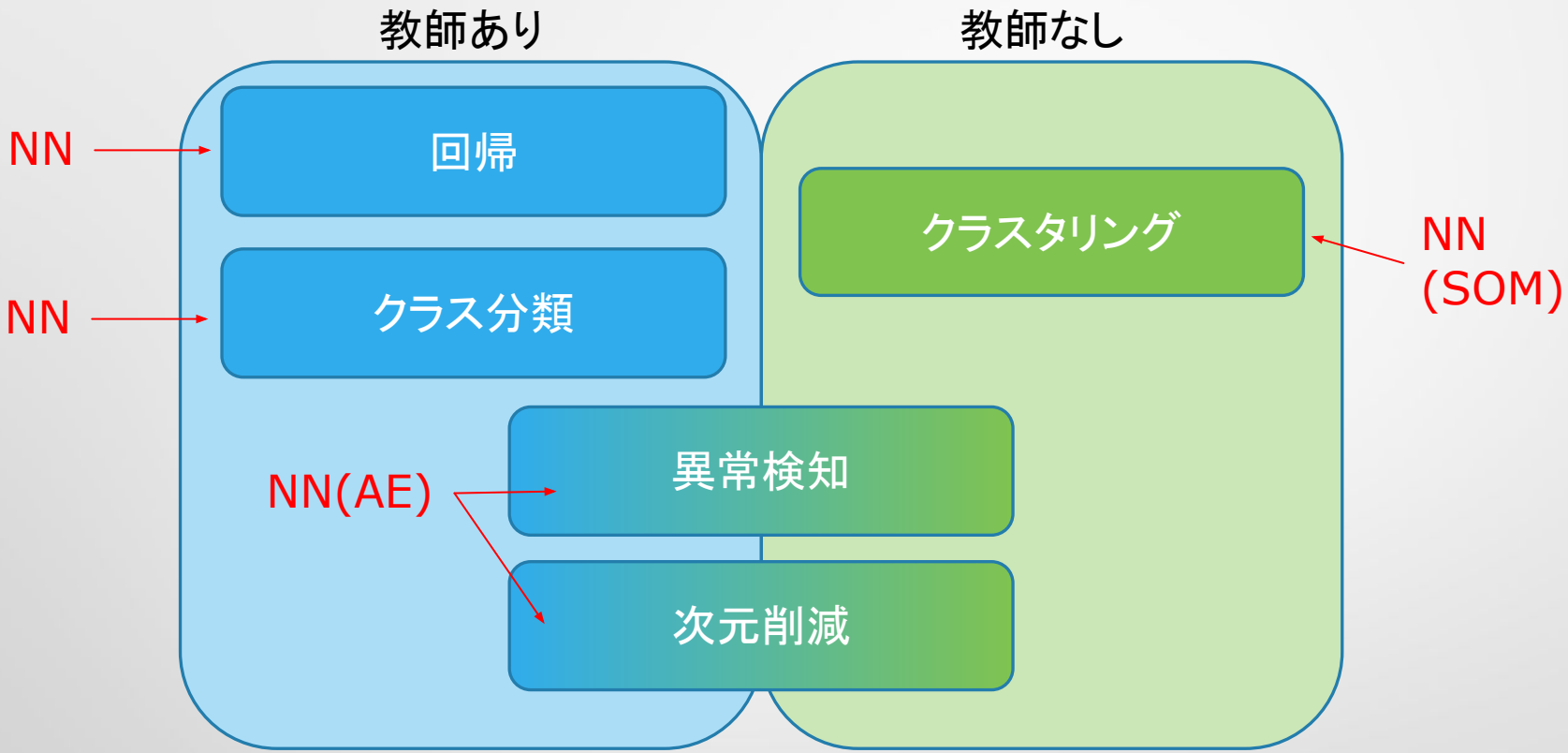
機械学習の手法・MNIST手書き文字認識

機械学習の手法



半教師あり学習
強化学習
転移学習

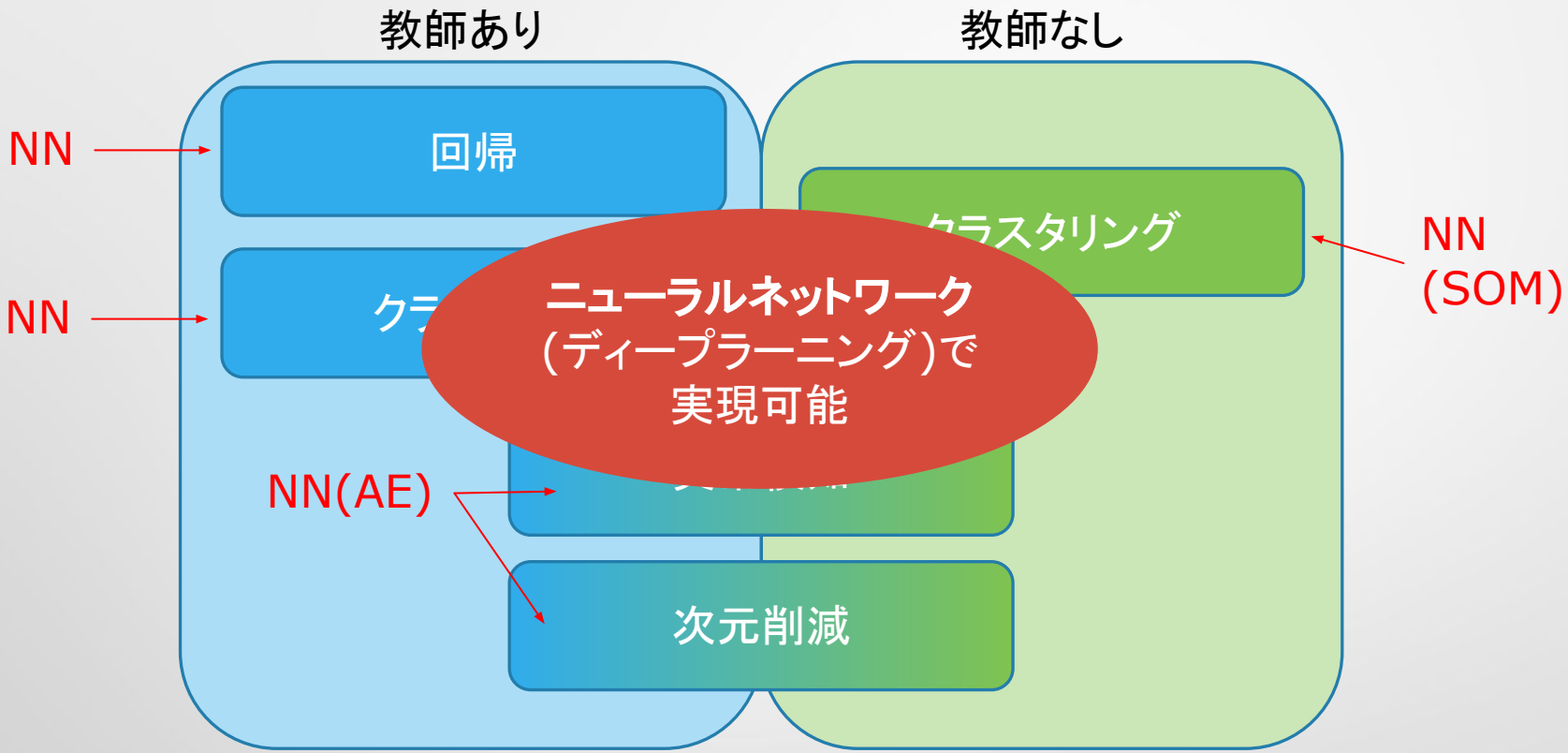
機械学習の手法



NN(GAN)
半教師あり学習
NN(DQL)
強化学習
転移学習
NN

※NN=ニューラルネット

機械学習の手法



NN(GAN)

半教師あり学習

NN(DQL)

強化学習

転移学習

NN

※NN=ニューラルネット

クラス分類の手法

教師あり

クラス分類

識別関数

最小二乗分類
k-NN
決定木
SVM
パーセプトロン

識別モデル

最尤推定
MAP推定
ロジスティック回帰
CRF
ニューラルネット

生成モデル

ナイーブベイズ
トピックモデル(LDA)
ニューラルネット

クラス分類の手法

教師あり

クラス分類

識別関数

最小二乗分類
k-NN
決定木
SVM
パーセプトロン

識別モデル

最尤推定
MAP推定
ロジスティック回帰
CRF
ニューラルネット

生成モデル

ナイーブベイズ
トピックモデル(LDA)
ニューラルネット

ニューラルネットワーク
（ディープラーニング）

今日やってみること

ディープラーニングで
”MNIST手書き文字判別”

- **手書き文字画像**を読み込んで,

書かれている**数字を判別するAI**

を作ってみる

クラス分類の手法

教師あり

クラス分類

識別関数

最小二乗分類
k-NN
決定木
SVM
パーセプトロン

識別モデル

最尤推定
MAP推定
ロジスティック回帰
CRF
ニューラルネット

ココ

生成モデル

ナイーブベイズ
トピックモデル(LDA)
ニューラルネット

ディープラーニングで
"MNIST手書き文字判別"

MNISTとは

- MNIST：機械学習用データセット

各数字6,000枚×10種類=60,000枚

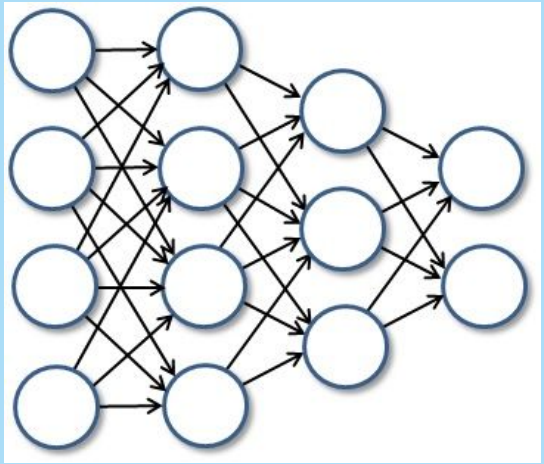
このデータを用いてAIを学習させる

MNIST手書き文字判別(1/2)

- AIを学習させる
- MNISTデータセットからAIが数字を学んでいく

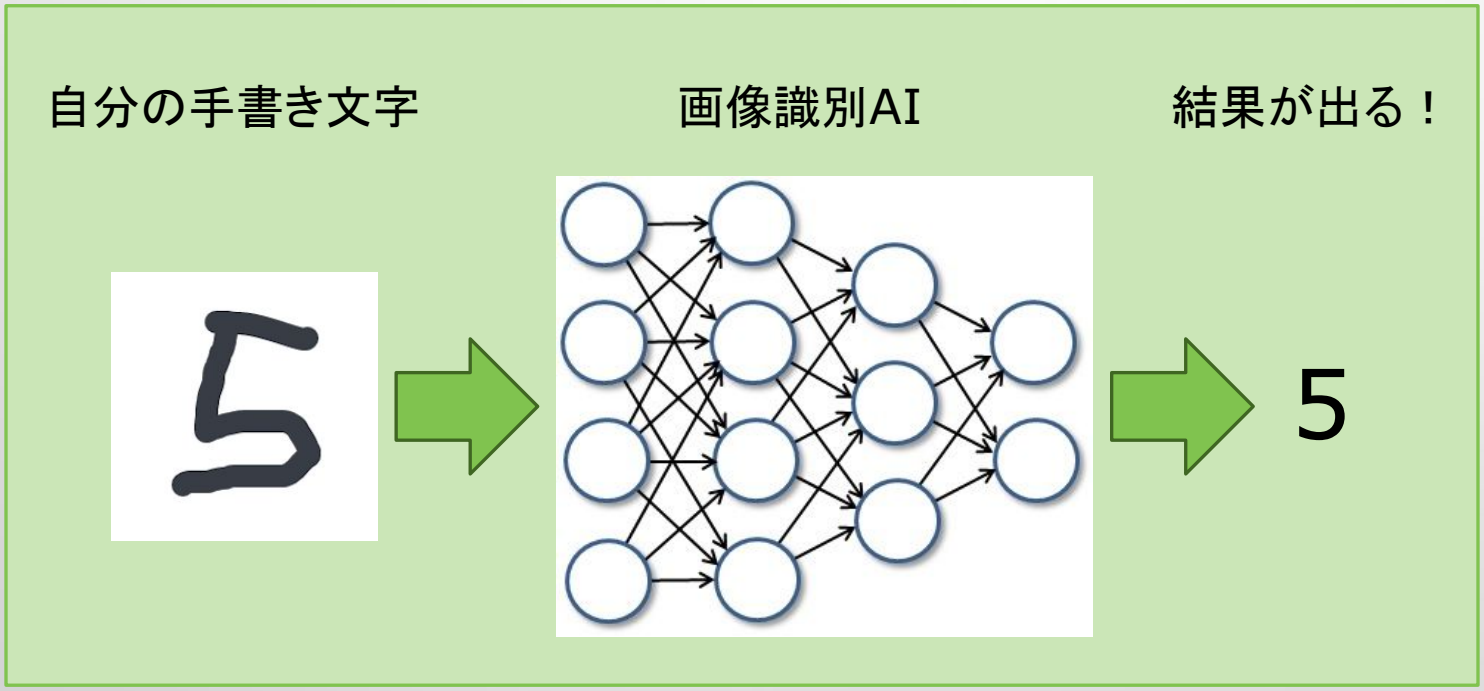
MNIST
データセット

画像識別AI



MNIST手書き文字判別(2/2)

- AIを使ってみる
- AIが自分で得た知識を元に，未知の画像を分類する



Hands on

MNIST手書き文字認識

Hands on

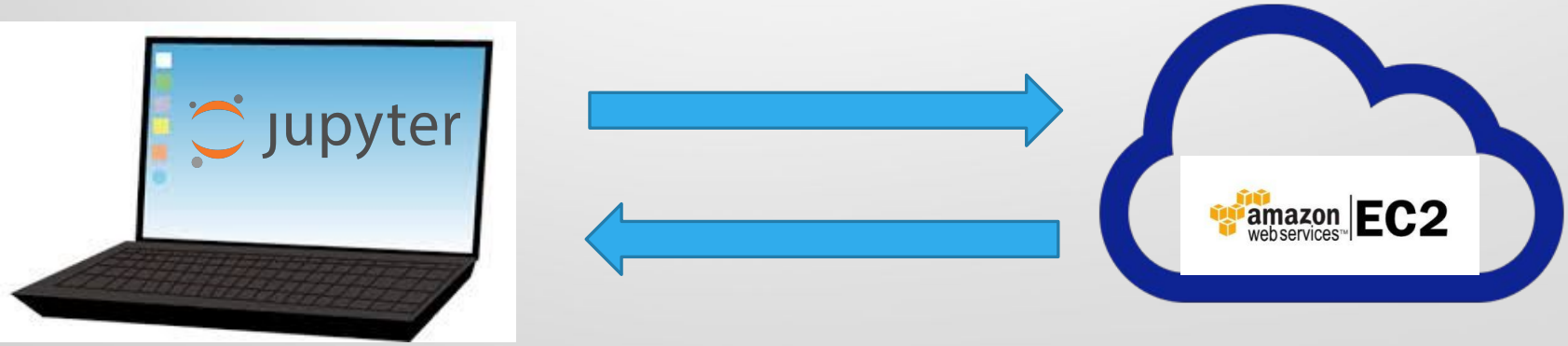
- **Amazon EC2 + Jupyter Notebook**
  - EC2でクラウド環境を用意
  - クラウド環境上でJupyter Notebookを実行

- **Python/Keras+TensorFlow**
  - Pythonは機械学習に強いプログラミング言語
  - Keras/TensorFlowの詳細は後述

Amazon EC2+Jupyter Notebook

- Amazon EC2
  - ➢ クラウドサーバ
- Jupyter Notebook
  - ➢ ノートブック形式のプログラム実行環境

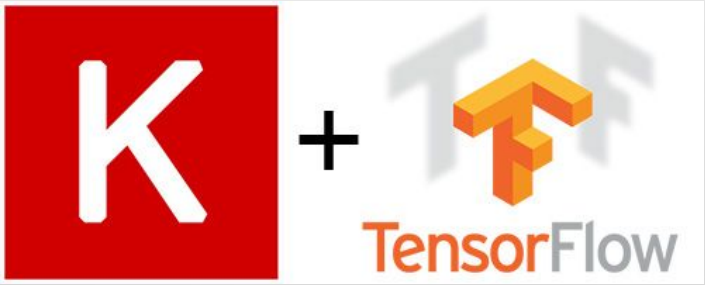
    メモ・実行結果を記録しながらコードが書ける
  - ➢ データサイエンティスト必須アイテム



Keras+TensorFlow

- TensorFlow
  - Google提供の深層学習用Pythonモジュール
  - Googleの各種サービスでも使われている

- Keras
  - TensorFlowをより直観的に呼び出せるラッパー
  - Modularity/Minimalism/Easy extensibility



Hands on List

1. 環境設定
   - インスタンス作成
   - ssh設定
   - 環境構築
   - BitBucket設定
   - Jupyter Notebook設定
2. サンプルプログラムの実行
3. コード解説
4. テスト

Hands on List

1. 環境設定
   - インスタンス作成
   - ssh設定
   - 環境構築
   - BitBucket設定
   - Jupyter Notebook設定
2. サンプルプログラムの実行
3. コード解説
4. テスト

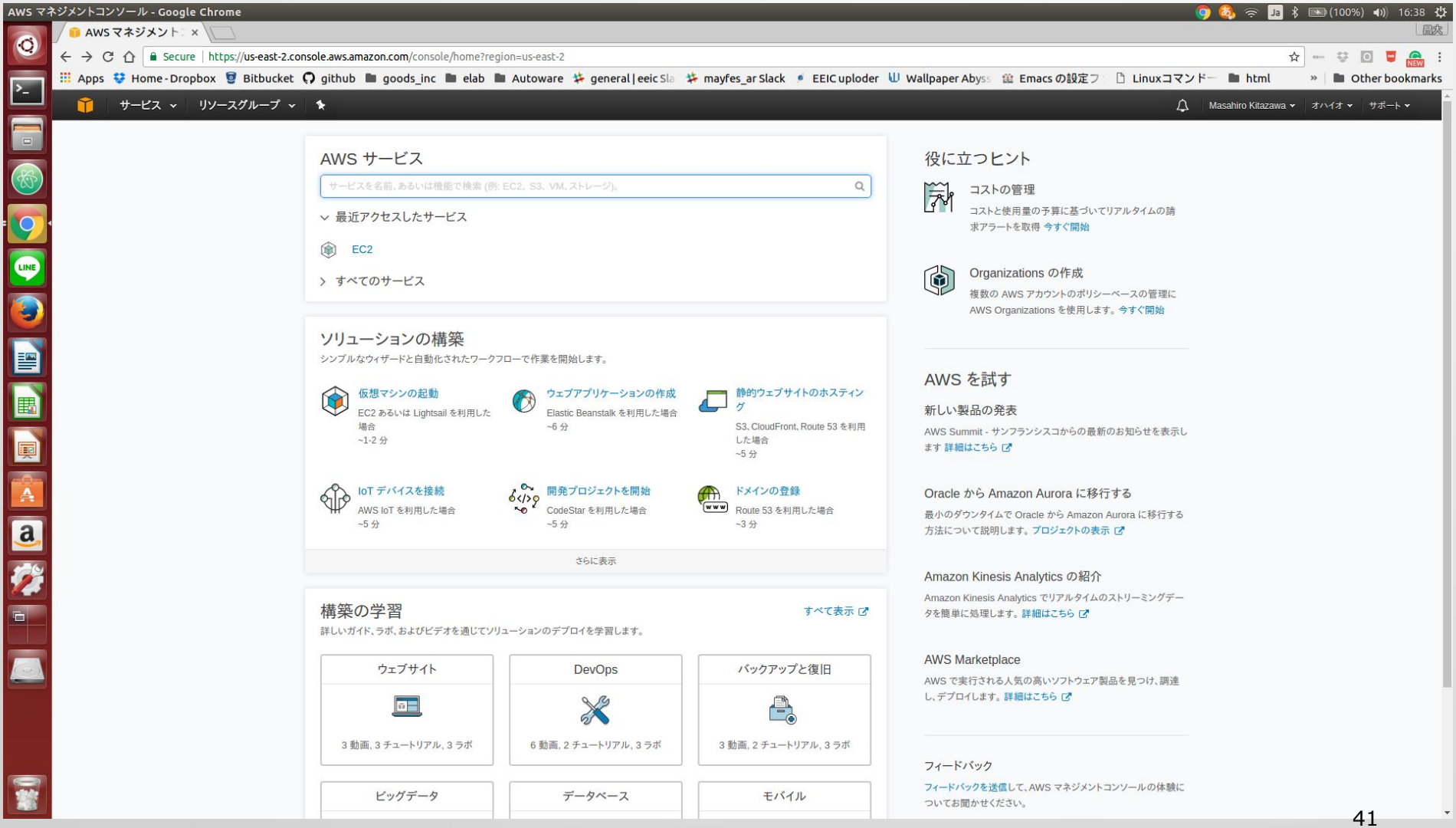
AWSアカウント作成

- まずAWSアカウントを作成してください
- https://aws.amazon.com/jp/

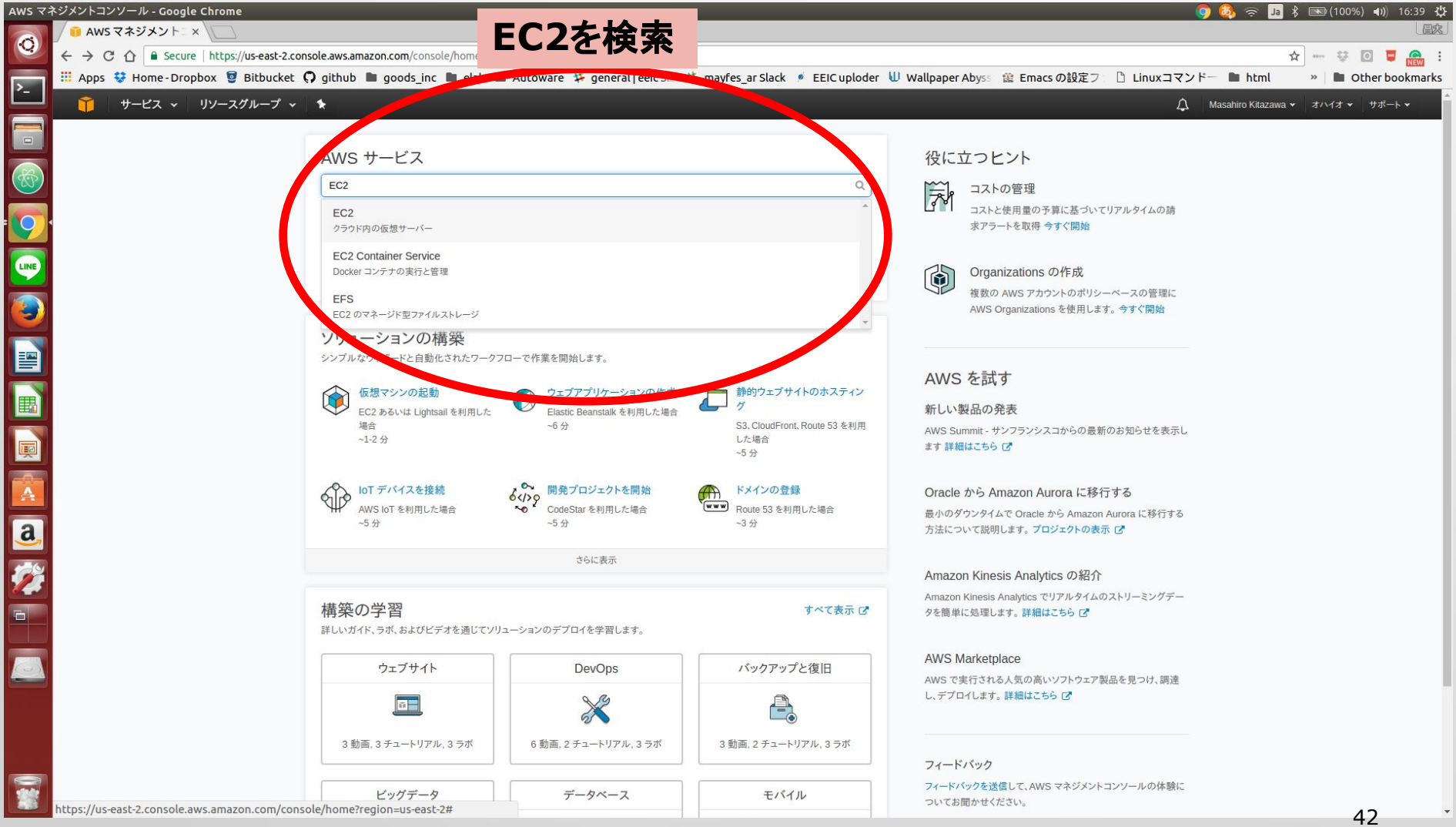
AWSアカウント作成

メニュー
amazon web services
無料利用枠
製品
ソリューション
料金モデル
お客様事例
詳細
日本語
アカウント
まずは無料で始める »
AWS の開始方法
アマゾン ウェブ サービスで構築を始めましょう
まずは無料で始める
AWS 無料利用枠の詳細はこちら »

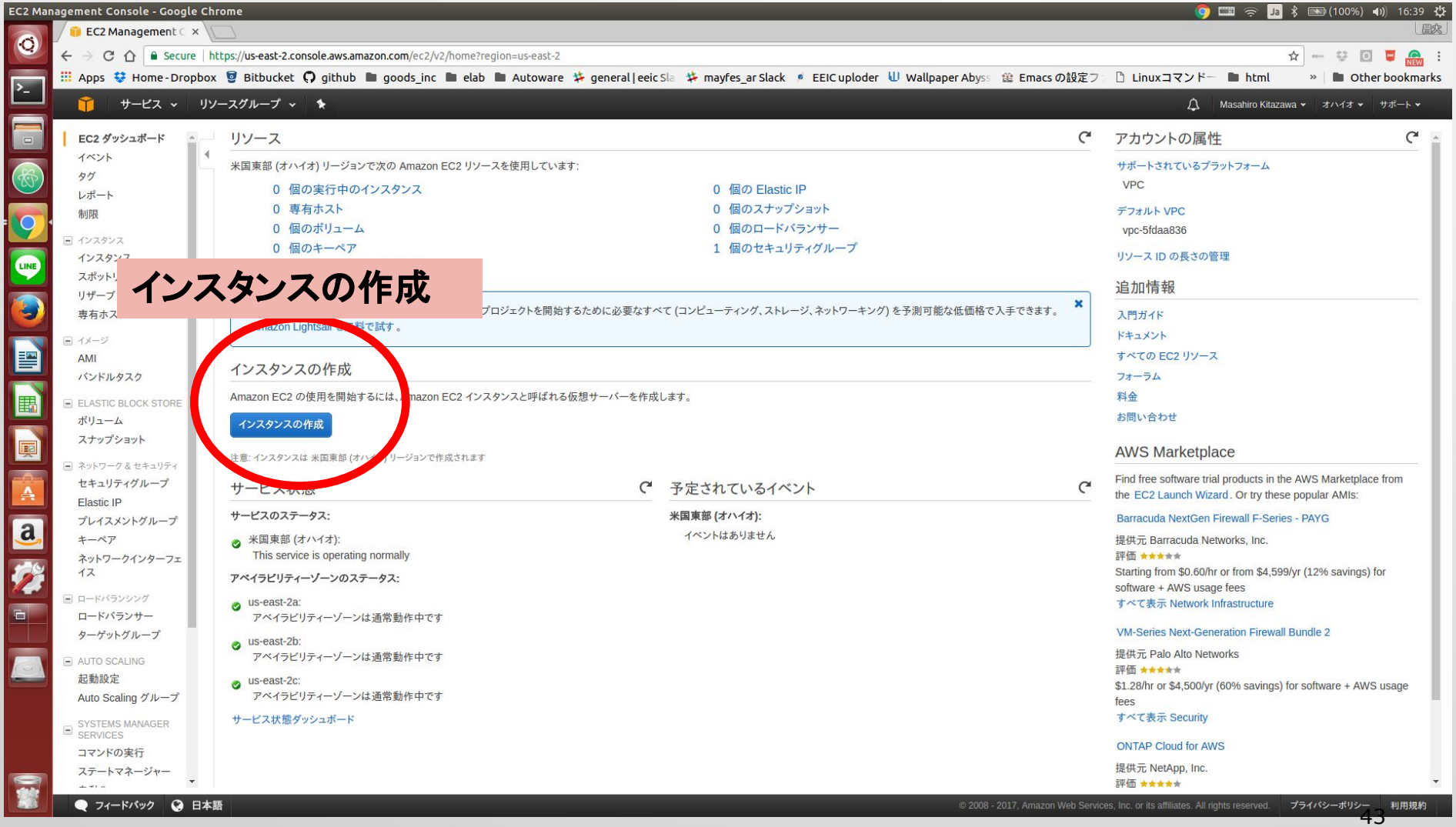
インスタンス作成


AWS マネジメントコンソール - Google Chrome
https://us-east-2.console.aws.amazon.com/console/home?region=us-east-2
サービス
リソースグループ
Masahiro Kitazawa
オハイオ
サポート
AWS サービス
サービスを名前、あるいは機能で検索 (例: EC2、S3、VM、ストレージ)。
最近アクセスしたサービス
EC2
すべてのサービス
ソリューションの構築
シンプルなウィザードと自動化されたワークフローで作業を開始します。
仮想マシンの起動
EC2 あるいは Lightsail を利用した場合
~1-2 分
ウェブアプリケーションの作成
Elastic Beanstalk を利用した場合
~6 分
静的ウェブサイトのホスティング
S3、CloudFront、Route 53 を利用した場合
~5 分
IoT デバイスを接続
AWS IoT を利用した場合
~5 分
開発プロジェクトを開始
CodeStar を利用した場合
~5 分
ドメインの登録
Route 53 を利用した場合
~3 分
さらに表示
構築の学習
すべて表示
詳しいガイド、ラボ、およびビデオを通じてソリューションのデプロイを学習します。
ウェブサイト
3 動画, 3 チュートリアル, 3 ラボ
DevOps
6 動画, 2 チュートリアル, 3 ラボ
バックアップと復旧
3 動画, 2 チュートリアル, 3 ラボ
ビッグデータ
データベース
モバイル
役に立つヒント
コストの管理
コストと使用量の予算に基づいてリアルタイムの請求アラートを取得 今すぐ開始
Organizations の作成
複数の AWS アカウントのポリシーベースの管理に AWS Organizations を使用します。今すぐ開始
AWS を試す
新しい製品の発表
AWS Summit - サンフランシスコからの最新のお知らせを表示します 詳細はこちら
Oracle から Amazon Aurora に移行する
最小のダウンタイムで Oracle から Amazon Aurora に移行する方法について説明します。プロジェクトの表示
Amazon Kinesis Analytics の紹介
Amazon Kinesis Analytics でリアルタイムのストリーミングデータを簡単に処理します。詳細はこちら
AWS Marketplace
AWS で実行される人気の高いソフトウェア製品を見つけ、調達し、デプロイします。詳細はこちら
フィードバック
フィードバックを送信して、AWS マネジメントコンソールの体験についてお聞かせください。

インスタンス作成


EC2を検索
AWS マネジメントコンソール - Google Chrome
https://us-east-2.console.aws.amazon.com/console/home
サービス
リソースグループ
Masahiro Kitazawa
オハイオ
サポート
AWS サービス
EC2
EC2
クラウド内の仮想サーバー
EC2 Container Service
Docker コンテナの実行と管理
EFS
EC2 のマネージド型ファイルストレージ
ソリューションの構築
仮想マシンの起動
EC2 あるいは Lightsail を利用した場合
~1-2 分
ウェブアプリケーションの作成
Elastic Beanstalk を利用した場合
~6 分
静的ウェブサイトのホスティング
S3、CloudFront、Route 53 を利用した場合
~5 分
IoT デバイスを接続
AWS IoT を利用した場合
~5 分
開発プロジェクトを開始
CodeStar を利用した場合
~5 分
ドメインの登録
Route 53 を利用した場合
~3 分
さらに表示
構築の学習
すべて表示
詳しいガイド、ラボ、およびビデオを通じてソリューションのデプロイを学習します。
ウェブサイト
3 動画, 3 チュートリアル, 3 ラボ
DevOps
6 動画, 2 チュートリアル, 3 ラボ
バックアップと復旧
3 動画, 2 チュートリアル, 3 ラボ
ビッグデータ
データベース
モバイル
役に立つヒント
コストの管理
コストと使用量の予算に基づいてリアルタイムの請求アラートを取得 今すぐ開始
Organizations の作成
複数の AWS アカウントのポリシーベースの管理に AWS Organizations を使用します。今すぐ開始
AWS を試す
新しい製品の発表
AWS Summit - サンフランシスコからの最新のお知らせを表示します 詳細はこちら
Oracle から Amazon Aurora に移行する
最小のダウンタイムで Oracle から Amazon Aurora に移行する方法について説明します。プロジェクトの表示
Amazon Kinesis Analytics の紹介
Amazon Kinesis Analytics でリアルタイムのストリーミングデータを簡単に処理します。詳細はこちら
AWS Marketplace
AWS で実行される人気の高いソフトウェア製品を見つけ、調達し、デプロイします。詳細はこちら
フィードバック
フィードバックを送信して、AWS マネジメントコンソールの体験についてお聞かせください。
https://us-east-2.console.aws.amazon.com/console/home?region=us-east-2#

インスタンス作成


EC2 Management Console - Google Chrome
https://us-east-2.console.aws.amazon.com/ec2/v2/home?region=us-east-2
サービス
リソースグループ
EC2 ダッシュボード
イベント
タグ
レポート
制限
インスタンス
スポットリ
リザーブ
専有ホス
イメージ
AMI
バンドルタスク
ELASTIC BLOCK STORE
ボリューム
スナップショット
ネットワーク & セキュリティ
セキュリティグループ
Elastic IP
プレイスメントグループ
キーペア
ネットワークインターフェイス
ロードバランシング
ロードバランサー
ターゲットグループ
AUTO SCALING
起動設定
Auto Scaling グループ
SYSTEMS MANAGER SERVICES
コマンドの実行
ステートマネージャー
インスタンスの作成
リソース
米国東部 (オハイオ) リージョンで次の Amazon EC2 リソースを使用しています:
0 個の実行中のインスタンス
0 専有ホスト
0 個のボリューム
0 個のキーペア
0 個の Elastic IP
0 個のスナップショット
0 個のロードバランサー
1 個のセキュリティグループ
インスタンスの作成
Amazon EC2 の使用を開始するには、Amazon EC2 インスタンスと呼ばれる仮想サーバーを作成します。
インスタンスの作成
サービス状態
サービスのステータス:
米国東部 (オハイオ):
This service is operating normally
アベイラビリティーゾーンのステータス:
us-east-2a:
アベイラビリティーゾーンは通常動作中です
us-east-2b:
アベイラビリティーゾーンは通常動作中です
us-east-2c:
アベイラビリティーゾーンは通常動作中です
サービス状態ダッシュボード
予定されているイベント
米国東部 (オハイオ):
イベントはありません
アカウントの属性
サポートされているプラットフォーム
VPC
デフォルト VPC
vpc-5fdaa836
リソース ID の長さの管理
追加情報
入門ガイド
ドキュメント
すべての EC2 リソース
フォーラム
料金
お問い合わせ
AWS Marketplace
Find free software trial products in the AWS Marketplace from the EC2 Launch Wizard. Or try these popular AMIs:
Barracuda NextGen Firewall F-Series - PAYG
提供元 Barracuda Networks, Inc.
評価
Starting from $0.60/hr or from $4,599/yr (12% savings) for software + AWS usage fees
すべて表示 Network Infrastructure
VM-Series Next-Generation Firewall Bundle 2
提供元 Palo Alto Networks
評価
$1.28/hr or $4,500/yr (60% savings) for software + AWS usage fees
すべて表示 Security
ONTAP Cloud for AWS
提供元 NetApp, Inc.
評価
フィードバック
日本語
© 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved.
プライバシーポリシー
利用規約

インスタンス作成

EC2 Management Console - Google Chrome

https://us-east-2.console.aws.amazon.com/ec2/v2/home?region=us-east-2#LaunchInstanceWizard:

1. AMI の選択 | 2. インスタンスタイプの選択 | 3. インスタンスの設定 | 4. ストレージの追加 | 5. タグの追加 | 6. セキュリティグループの設定 | 7. 確認

ステップ 1: Amazon マシンイメージ (AMI)

AMI は、インスタンスの作成に必要なソフトウェア構成 (OS、アプリケーションサーバー、アプリケーション) を含むテンプレートです。AMI は、AWS が提供するもの、ユーザーコミュニティが提供するもの、または AWS Marketplace に掲載されているものを選択できます。独自の AMI のいずれかを選択することもできます。

キャンセルして終了

クイックスタート

マイ AMI
AWS Marketplace
コミュニティ AMI
無料利用枠のみ

32 個のAMI 中 1 ~ 32

Amazon Linux AMI 2017.03.1 (HVM), SSD Volume Type - ami-8a7859ef
The Amazon Linux AMI is an EBS-backed, AWS-supported image. The default image includes AWS command line tools, Python, Ruby, Perl, and Java. The repositories include Docker, PHP, MySQL, PostgreSQL, and other packages.
ルートデバイスタイプ: ebs 仮想化タイプ: hvm
Amazon Linux 無料利用枠の対象 | 選択 | 64 ビット

SUSE Linux Enterprise Server 12 SP2 (HVM), SSD Volume Type - ami-61a7fd04
SUSE Linux Enterprise Server 12 Service Pack 2 (HVM), EBS Gen … Scripting, and Legacy modules enabled.
ルートデバイスタイプ: ebs 仮想化タイプ: hvm
SUSE Linux 無料利用枠の対象 | 選択 | 64 ビット

OSの選択(Ubuntu)

Red Hat Enterprise Linux 7.3 (HVM), SSD Volume Type - ami-11aa8c74
Red Hat Enterprise Linux version 7.3 … General Purpose (SSD) Volume Type
ルートデバイスタイプ: ebs 仮想化タイプ: hvm
Red Hat 無料利用枠の対象 | 選択 | 64 ビット

Ubuntu Server 16.04 LTS (HVM), SSD Volume Type - ami-8b92b4ee
Ubuntu Server 16.04 LTS (HVM),EBS General Purpose (SSD) Volume Type. Support available from Canonical (http://www.ubuntu.com/cloud/services).
ルートデバイスタイプ: ebs 仮想化タイプ: hvm
無料利用枠の対象 | 選択 | 64 ビット

Microsoft Windows Server 2016 Base - ami-3487a151
Microsoft Windows 2016 Datacenter edition. [English]
ルートデバイスタイプ: ebs 仮想化タイプ: hvm
Windows 無料利用枠の対象 | 選択 | 64 ビット

データベースインスタンスを起動中ですか。Amazon RDS をお試しください。 非表示
Amazon Relational Database Service (RDS) では、時間のかかるデータベース管理タスクを自動化することにより、AWS で簡単にデータベースをセットアップ、運用、およびスケーリングできます。RDS を利用すると、**Amazon Aurora、MariaDB、MySQL、Oracle、PostgreSQL および SQL サーバー**データベースを AWS で簡単にデプロイできます。Aurora は、MySQL 互換の、エンタープライズ級データベースで、コストは商用データベースの 10 分の 1 です。RDS の詳細
Amazon RDS
RDS を使いデータベースを起動

フィードバック | 日本語 | © 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved. | プライバシーポリシー | 利用規約

インスタンス作成

EC2 Management Console - Google Chrome

https://us-east-2.console.aws.amazon.com/ec2/v2/home?region=us-east-2#LaunchInstanceWizard:

1. AMI の選択 | 2. インスタンスタイプの選択 | 3. インスタンスの設定 | 4. ストレージの追加 | 5. タグの追加 | 6. セキュリティグループの設定 | 7. 確認

ステップ 2: インスタンスタイプの選択

Amazon EC2 では、異なるユースケースに合わせて最適化されたさまざまなインスタンスタイプが用意されています。インスタンスは、アプリケーションを実行できる仮想サーバーです。CPU、メモリ、ストレージ、ネットワークキャパシティーのさまざまな組み合わせが可能なため、アプリケーションに合わせて適切なリソースを柔軟に選択できます。インスタンスタイプおよびそれをコンピューティングのニーズに適用する方法に関する 詳細はこちら。

フィルター条件: すべてのインスタンスタイプ | 現行世代 | 列の表示/非表示

現在選択中: t2.micro (可変 ECU, 1 vCPU, 2.5 GHz, Intel Xeon Family, 1 GiB メモリ, EBS のみ)

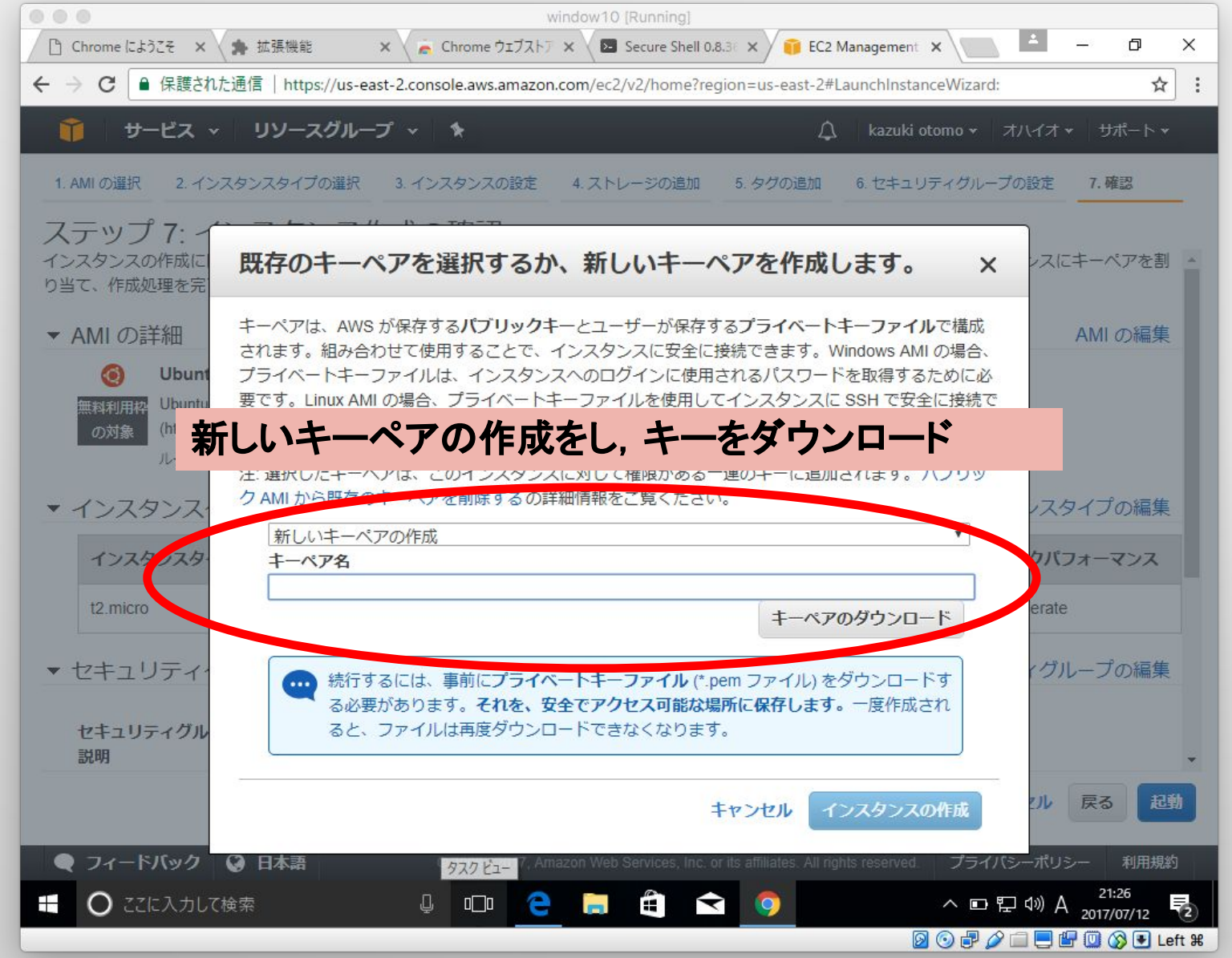
L2micro (デフォルト)

	ファミリー	タイプ	vCPU			EBS 最適化利用	ネットワークパフォーマンス	IPv6 サポート
	[illegible]	t2.nano	1	0.5	EBS のみ	-	[illegible]	はい
■	汎用	t2.micro 無料利用枠の対象	1	1	EBS のみ	-	低から中	はい
	[illegible]	t2.small	1	2	EBS のみ	-	[illegible]	はい
	汎用	t2.medium	2	4	EBS のみ	-	低から中	はい
	汎用	t2.large	2	8	EBS のみ	-	低から中	はい
	汎用	t2.xlarge	4	16	EBS のみ	-	中	はい
	汎用	t2.2xlarge	8	32	EBS のみ	-	中	はい
	汎用	m4.large	2	8	EBS のみ	はい	中	はい
	汎用	m4.xlarge	4	16	EBS のみ	はい	高い	はい
	汎用	m4.2xlarge	8	32	EBS のみ	はい	高い	はい
	汎用	m4.4xlarge	16	64	EBS のみ	はい	高い	はい
	汎用	m4.10xlarge	40	160	EBS のみ	はい	10 ギガビット	はい
	汎用	m4.16xlarge	64	256	EBS のみ	はい	20 ギガビット	はい

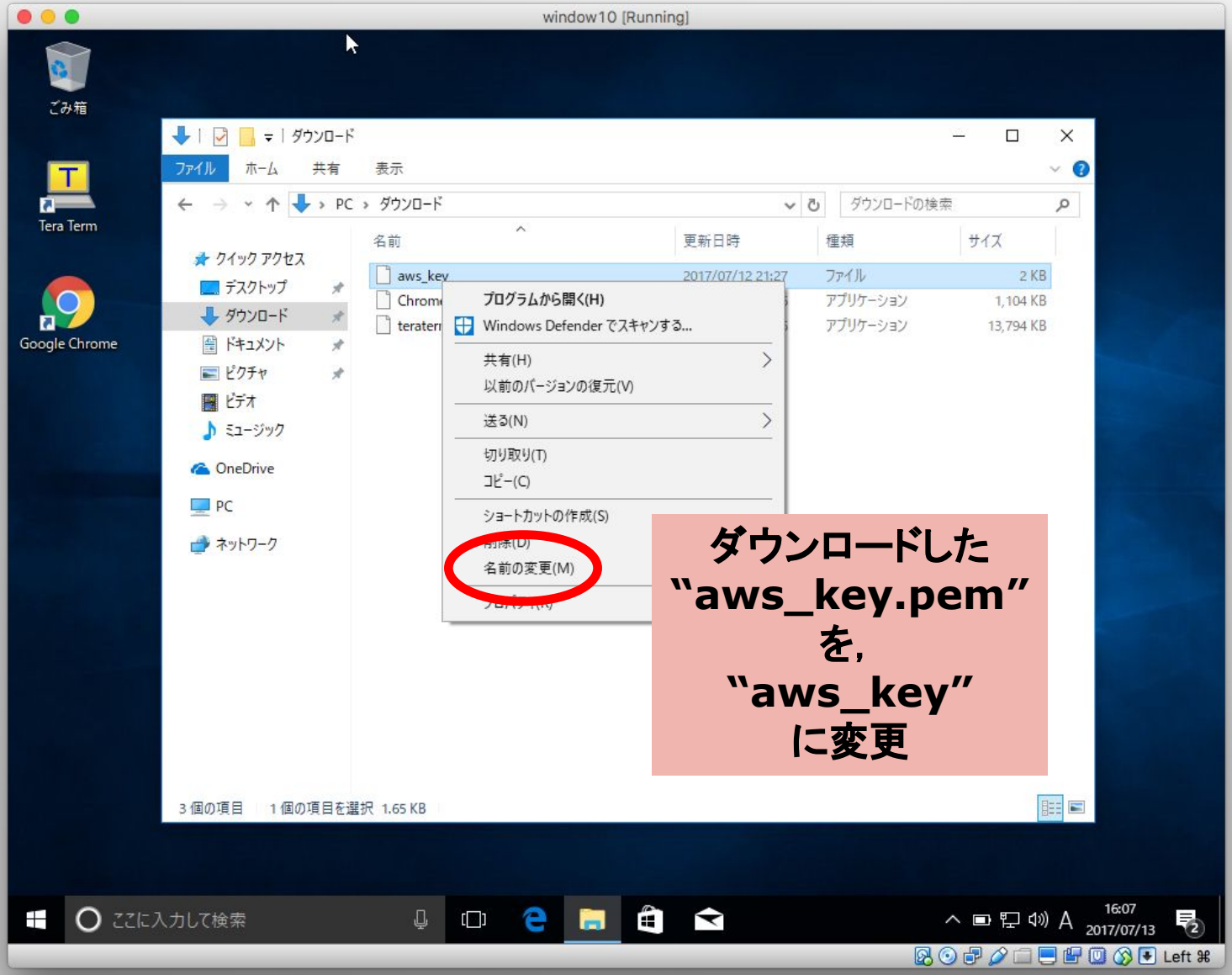
キャンセル | 戻る | 確認と作成 | 次の手順: インスタンスの詳細の設定

© 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved.

インスタンス作成


window10 [Running]
Chrome にようこそ
拡張機能
Chrome ウェブストア
Secure Shell 0.8.36
EC2 Management
保護された通信 | https://us-east-2.console.aws.amazon.com/ec2/v2/home?region=us-east-2#LaunchInstanceWizard:
サービス
リソースグループ
kazuki otomo
オハイオ
サポート
1. AMI の選択
2. インスタンスタイプの選択
3. インスタンスの設定
4. ストレージの追加
5. タグの追加
6. セキュリティグループの設定
7. 確認
既存のキーペアを選択するか、新しいキーペアを作成します。
キーペアは、AWS が保存するパブリックキーとユーザーが保存するプライベートキーファイルで構成されます。組み合わせて使用することで、インスタンスに安全に接続できます。Windows AMI の場合、プライベートキーファイルは、インスタンスへのログインに使用されるパスワードを取得するために必要です。Linux AMI の場合、プライベートキーファイルを使用してインスタンスに SSH で安全に接続で
新しいキーペアの作成をし，キーをダウンロード
新しいキーペアの作成
キーペア名
キーペアのダウンロード
続行するには、事前にプライベートキーファイル (*.pem ファイル) をダウンロードする必要があります。それを、安全でアクセス可能な場所に保存します。一度作成されると、ファイルは再度ダウンロードできなくなります。
キャンセル
インスタンスの作成
AMI の詳細
AMI の編集
t2.micro
戻る
起動
フィードバック
日本語
プライバシーポリシー
利用規約
ここに入力して検索
21:26
2017/07/12

インスタンス作成



ダウンロードした
“aws_key.pem”
を,
“aws_key”
に変更

インスタンス作成

EC2 Management Console - Google Chrome

https://us-east-2.console.aws.amazon.com/ec2/v2/home?region=us-east-2#LaunchInstanceWizard:

サービス　リソースグループ　Masahiro Kitazawa　オハイオ　サポート

作成ステータス

インスタンスは現在作成中です
次のインスタンスの作成が開始されました: i-0a45468243a2df5ed　作成ログの表示

予想請求額の通知を受け取る
請求アラートの作成 AWS 請求書の予想請求額が設定した金額を超えた場合 (つまり、無料利用枠を超えた場合) 、メール通知を受け取ります。

インスタンスへの接続方法

インスタンスは作成中です。**実行中**状態になり、使用する準備ができるまでに数分かかることがあります。新しいインスタンスの使用時間は、すぐに始まり、インスタンスを停止または削除するまで継続します。

[**インスタンスの表示**] をクリックして、インスタンスのステータスを監視します。インスタンスが一度**実行中**状態になれば、[インスタンス] 画面から**接続**できます。インスタンスへの接続方法の 詳細はこちら。

ここには、作業を始めるのに役立つリソースがあります

- Linux インスタンスへの接続方法
- AWS 無料利用枠の詳細
- Amazon EC2: ユーザーガイド
- Amazon EC2: ディスカッションフォーラム

インスタンスの作成中、次のことも行うことができます

ステータスチェックアラームの作成 これらのインスタンスがステータスチェックに合格しなかったときは通知が送信されます。(追加料金が適用される場合があります)
追加の EBS ボリュームを作成してアタッチする（追加料金が適用される場合があります)
セキュリティグループの管理

インスタンスの表示

インスタンスの表示

フィードバック　日本語　© 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved.　プライバシーポリシー　利用規約

aws_key.pem　Show all

インスタンス作成

EC2 Management Console - Google Chrome

https://us-east-2.console.aws.amazon.com/ec2/v2/home?region=us-east-2#Instances:sort=instanceId

インスタンスの作成 | 接続 | アクション

Name	インスタンス ID	インスタンスタイプ	アベイラビリティーゾ	インスタンスの状	ステータスチェック	アラームのステー	パブリック DNS (IPv4)	IPv4 パブリック IP	IPv6 IP	キー名	モニタリング
	i-0a45468243a2df5ed	t2.micro	us-east-2a	running	2/2 のチェッ...	なし	ec2-13-59-33-30.us-ea...	13.59.33.30	-	aws_key	disabled

インスタンス: i-0a45468243a2df5ed　パブリック DNS: ec2-13-59-33-30.us-east-2.compute.amazonaws.com

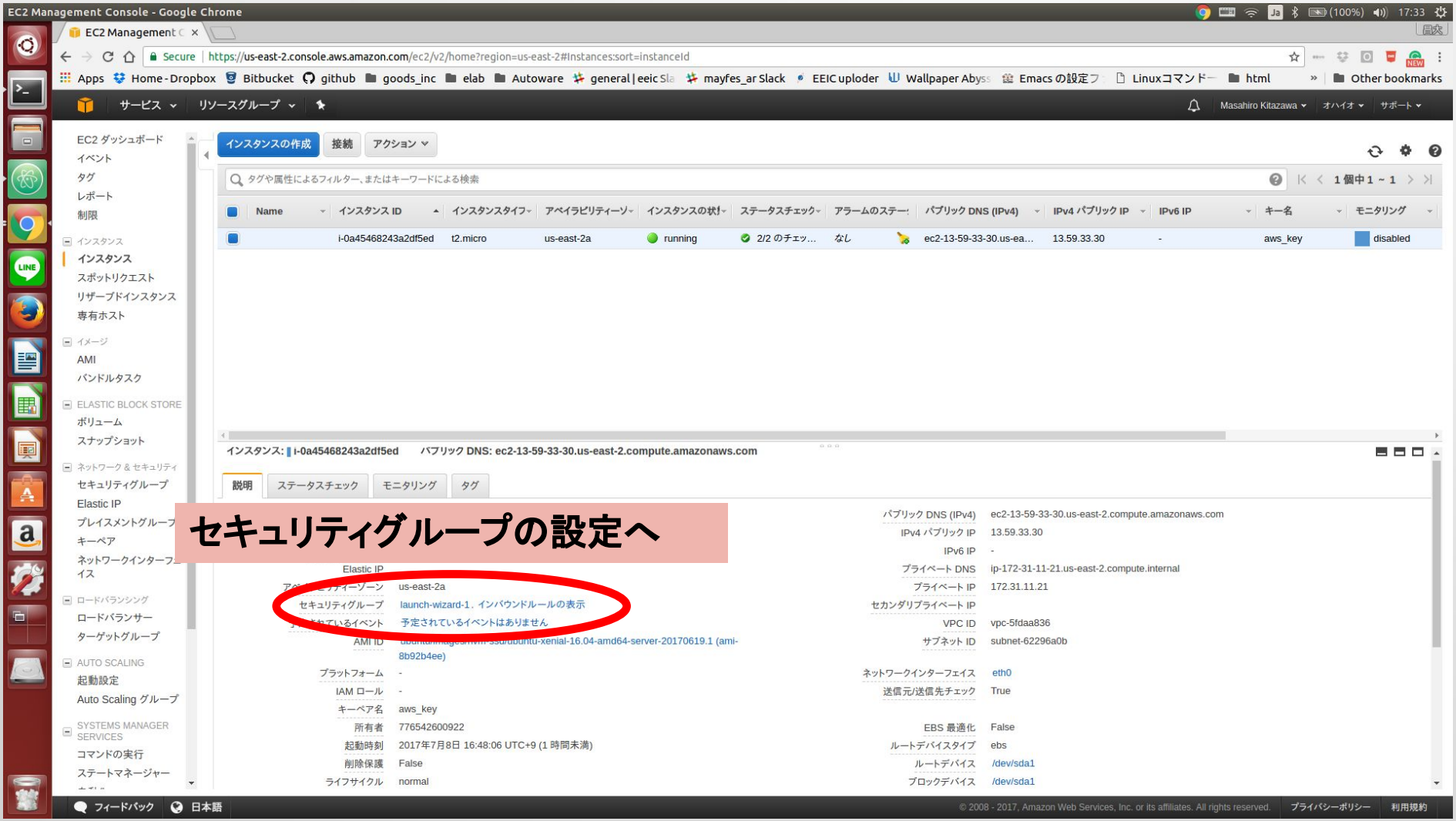
説明 | ステータスチェック | モニタリング | タグ

インスタンス ID	i-0a45468243a2df5ed
インスタンスの状態	running
インスタンスタイプ	t2.micro
Elastic IP	
アベイラビリティーゾーン	us-east-2a
セキュリティグループ	launch-wizard-1. インバウンドルールの表示
予定されているイベント	予定されているイベントはありません
AMI ID	ubuntu/images/hvm-ssd/ubuntu-xenial-16.04-amd64-server-20170619.1 (ami-8b92b4ee)
プラットフォーム	-
IAM ロール	-
キーペア名	aws_key
所有者	776542600922
起動時刻	2017年7月8日 16:48:06 UTC+9 (1 時間未満)
削除保護	False
ライフサイクル	normal

パブリック DNS (IPv4)	ec2-13-59-33-30.us-east-2.compute.amazonaws.com
IPv6 IP	-
プライベート DNS	ip-172-31-11-21.us-east-2.compute.internal
プライベート IP	172.31.11.21
セカンダリプライベート IP	
VPC ID	vpc-5fdaa836
サブネット ID	subnet-62296a0b
ネットワークインターフェイス	eth0
送信元/送信先チェック	True
EBS 最適化	False
ルートデバイスタイプ	ebs
ルートデバイス	/dev/sda1
ブロックデバイス	/dev/sda1

パブリックDNS をメモ(後で使います)

インスタンス作成


EC2 Management Console - Google Chrome
https://us-east-2.console.aws.amazon.com/ec2/v2/home?region=us-east-2#Instances:sort=instanceId
インスタンスの作成
接続
アクション
Name
インスタンス ID
i-0a45468243a2df5ed
t2.micro
us-east-2a
running
2/2 のチェッ...
なし
ec2-13-59-33-30.us-ea...
13.59.33.30
aws_key
disabled
インスタンス: i-0a45468243a2df5ed
パブリック DNS: ec2-13-59-33-30.us-east-2.compute.amazonaws.com
説明
ステータスチェック
モニタリング
タグ
セキュリティグループの設定へ
Elastic IP
us-east-2a
セキュリティグループ
launch-wizard-1. インバウンドルールの表示
予定されているイベントはありません
AMI ID
プラットフォーム
IAM ロール
キーペア名
aws_key
所有者
776542600922
起動時刻
2017年7月8日 16:48:06 UTC+9 (1 時間未満)
削除保護
False
ライフサイクル
normal
パブリック DNS (IPv4)
ec2-13-59-33-30.us-east-2.compute.amazonaws.com
IPv4 パブリック IP
13.59.33.30
IPv6 IP
プライベート DNS
ip-172-31-11-21.us-east-2.compute.internal
プライベート IP
172.31.11.21
セカンダリプライベート IP
VPC ID
vpc-5fdaa836
サブネット ID
subnet-62296a0b
ネットワークインターフェイス
eth0
送信元/送信先チェック
True
EBS 最適化
False
ルートデバイスタイプ
ebs
ルートデバイス
/dev/sda1
ブロックデバイス
/dev/sda1

インスタンス作成

EC2 Management Console - Google Chrome

https://us-east-2.console.aws.amazon.com/ec2/v2/home?region=us-east-2#SecurityGroups:groupId=sg-811b0de8;sort=groupId

セキュリティグループの作成 | アクション

グループ ID : sg-811b0de8 フィルターの追加

1 個中 1 ~ 1

Name	グループ ID	グループ名	VPC ID	説明
	sg-811b0de8	launch-wizard-1	vpc-5fdaa836	launch-wizard-1 created 2017-07-08T16:45:52.558+09:00

セキュリティグループ: sg-811b0de8

説明 | インバウンド | アウトバウンド | タグ

編集

編集をクリック

タイプ	プロトコル	ポート範囲	ソース
HTTP	TCP	80	0.0.0.0/0
HTTP	TCP	80	::/0
カスタム TCP ルール	TCP	8888	0.0.0.0/0
カスタム TCP ルール	TCP	8888	::/0
SSH	TCP	22	0.0.0.0/0

© 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved.

インスタンス作成

EC2 Management Console - Google Chrome
https://us-east-2.console.aws.amazon.com/ec2/v2/home?region=us-east-2#SecurityGroups:groupId=sg-811b0de8;sort=groupId
セキュリティグループの作成
アクション
カスタムTCP
ポート:8888
ソース 0.0.0.0/0
::/0
インバウンドのルールの編集
タイプ
プロトコル
ポート範囲
ソース
カスタム TCP ルール
TCP
8888
0.0.0.0/0
::/0
SSH
22
ルールの追加
注意: 既存のルールを編集すると、編集したルールが削除されて、新しい詳細を含む新しいルールが作成されます。これにより、そのルールに依存するトラフィックは、新しいルールが作成されるまで非常に短時間切断されます。
キャンセル
保存
HTTP
80

Hands on List

1. 環境設定
   - インスタンス作成
   - ssh設定
   - 環境構築
   - BitBucket設定
   - Jupyter Notebook設定
2. サンプルプログラムの実行
3. コード解説
4. テスト

ssh設定

- インスタンスに接続し,
  コンソール表示できるようにしましょう

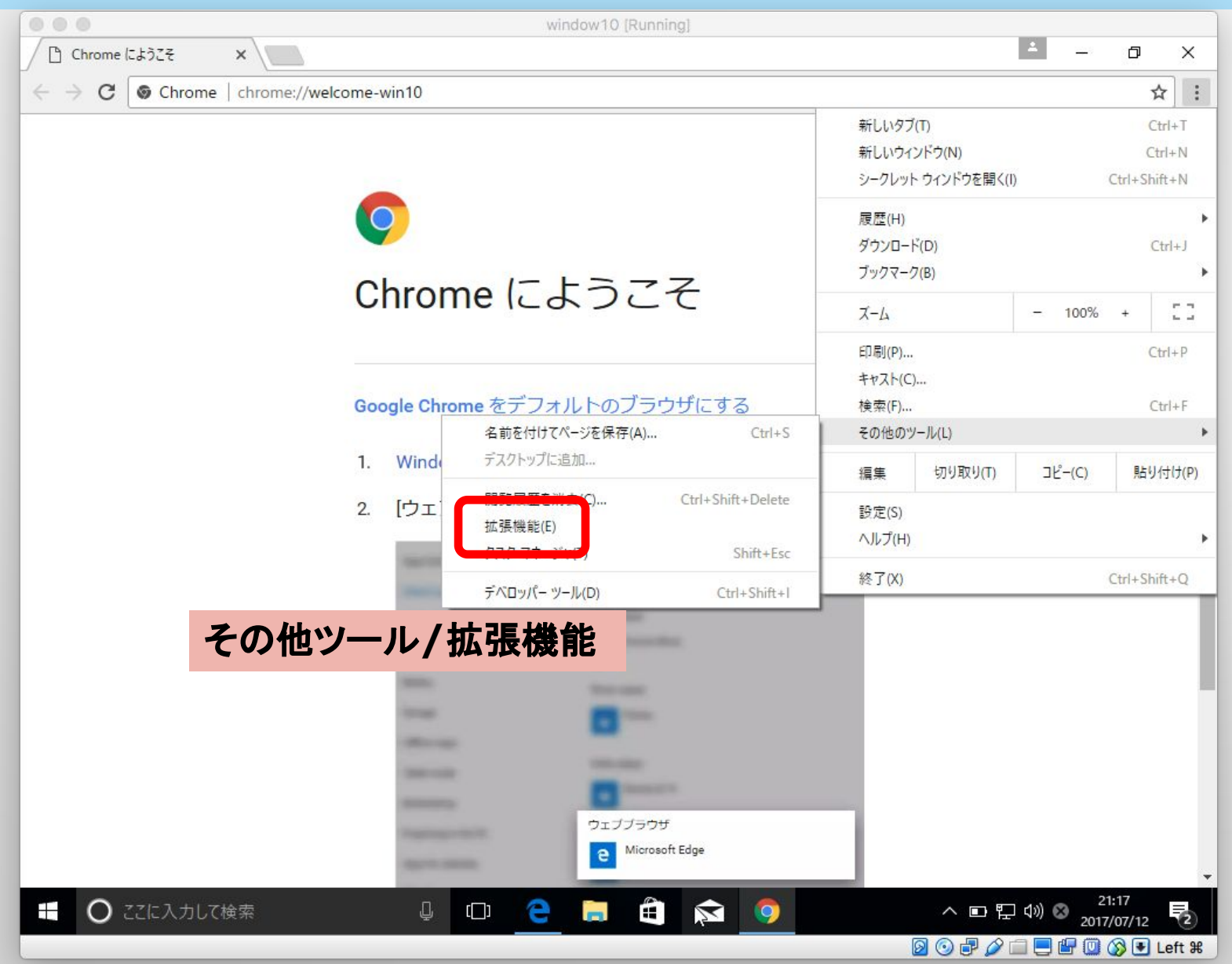
- Chrome Secure Shellを使います
  - MacOSの方はターミナル推奨

- その他環境(Terra termなど)
  - http://dev.classmethod.jp/cloud/aws/aws-beginner-ec2-ssh/

ssh設定

- まずChromeを起動してください
- ない方はこちらから
  - https://www.google.co.jp/chrome/browser/desktop/index.html

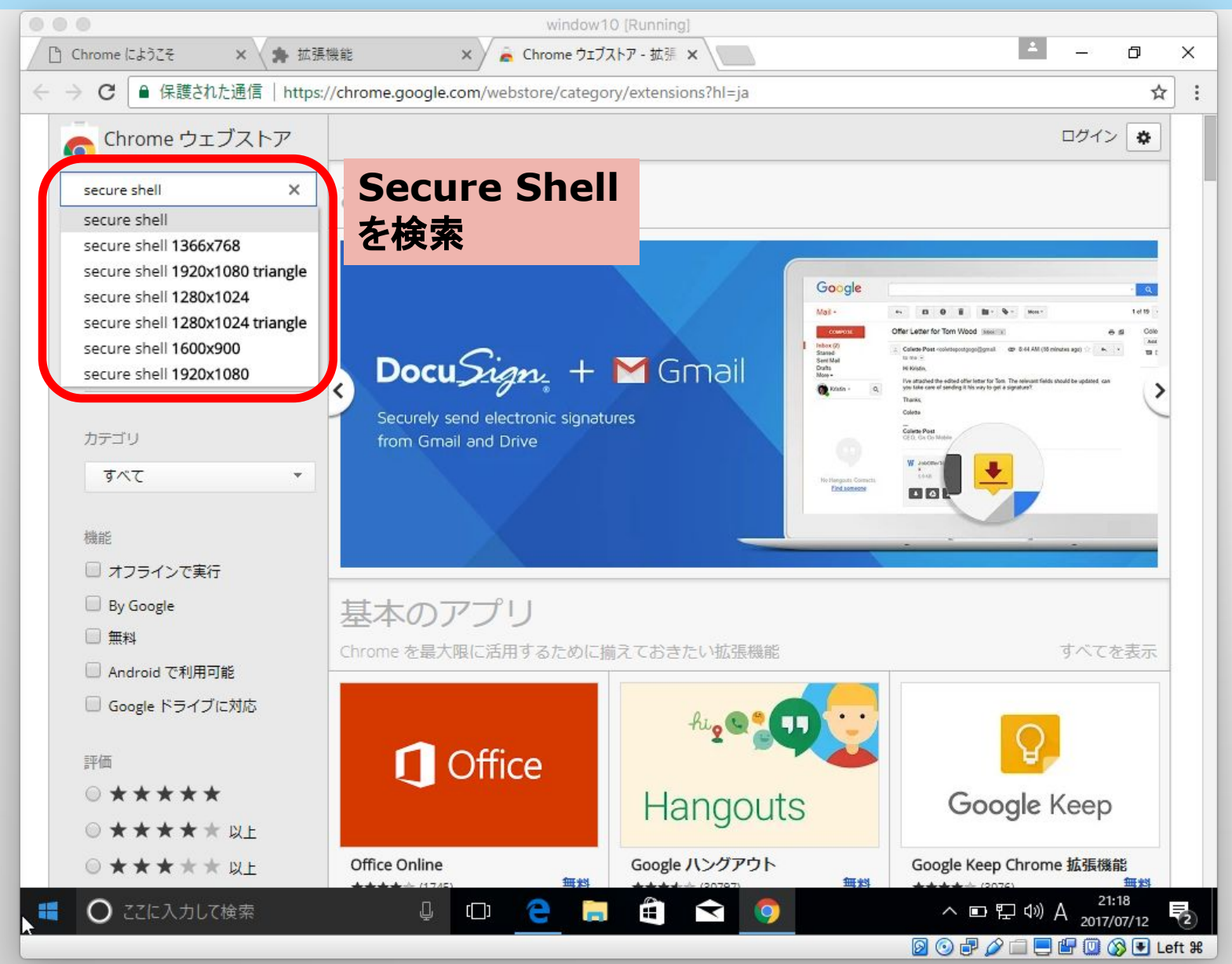
ssh設定


window10 [Running]
Chrome ようこそ
Chrome
chrome://welcome-win10
Chrome ようこそ
Google Chrome をデフォルトのブラウザにする
新しいタブ(T) Ctrl+T
新しいウィンドウ(N) Ctrl+N
シークレット ウィンドウを開く(I) Ctrl+Shift+N
履歴(H)
ダウンロード(D) Ctrl+J
ブックマーク(B)
ズーム − 100% +
印刷(P)... Ctrl+P
キャスト(C)...
検索(F)... Ctrl+F
その他のツール(L)
編集 切り取り(T) コピー(C) 貼り付け(P)
設定(S)
ヘルプ(H)
終了(X) Ctrl+Shift+Q
名前を付けてページを保存(A)... Ctrl+S
デスクトップに追加...
拡張機能(E)
Shift+Esc
デベロッパー ツール(D) Ctrl+Shift+I
その他ツール/拡張機能
ウェブブラウザ
Microsoft Edge
ここに入力して検索
21:17
2017/07/12
Left ⌘

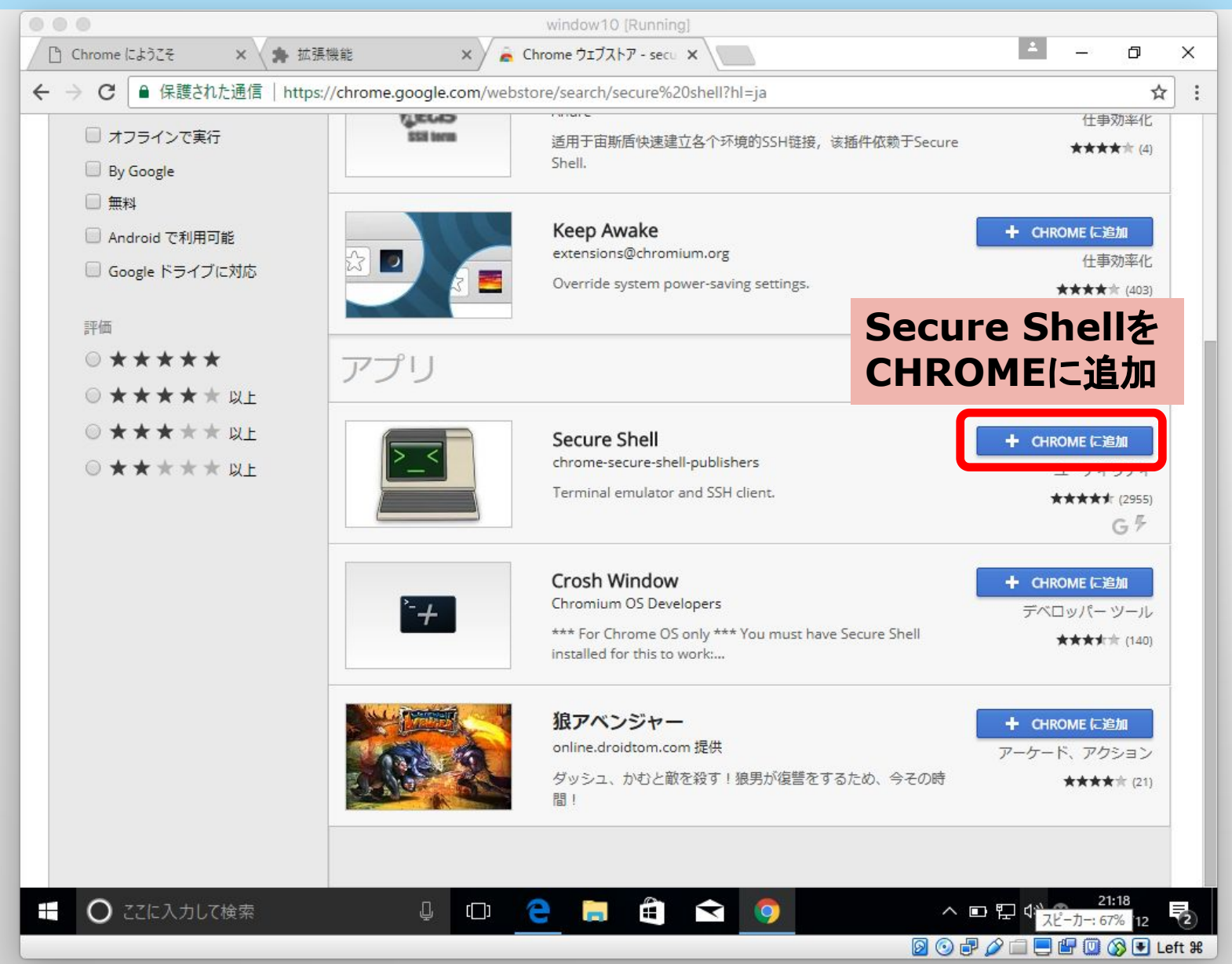
ssh設定

window10 [Running]
Chrome にようこそ
拡張機能
Chrome
chrome://extensions
Chrome
拡張機能
拡張機能
デベロッパー モード
シークレット モードでの実行を許可する
Google スプレッドシート 1.1
有効
スプレッドシートの作成と編集
詳細
シークレット モードでの実行を許可する
Google スライド 0.9
有効
プレゼンテーションの作成と編集
詳細
シークレット モードでの実行を許可する
Google ドキュメント 0.9
有効
文書の作成と編集
詳細
シークレット モードでの実行を許可する
他の拡張機能を見る
他の拡張機能を見る
ここに入力して検索
21:17
2017/07/12
Left ⌘

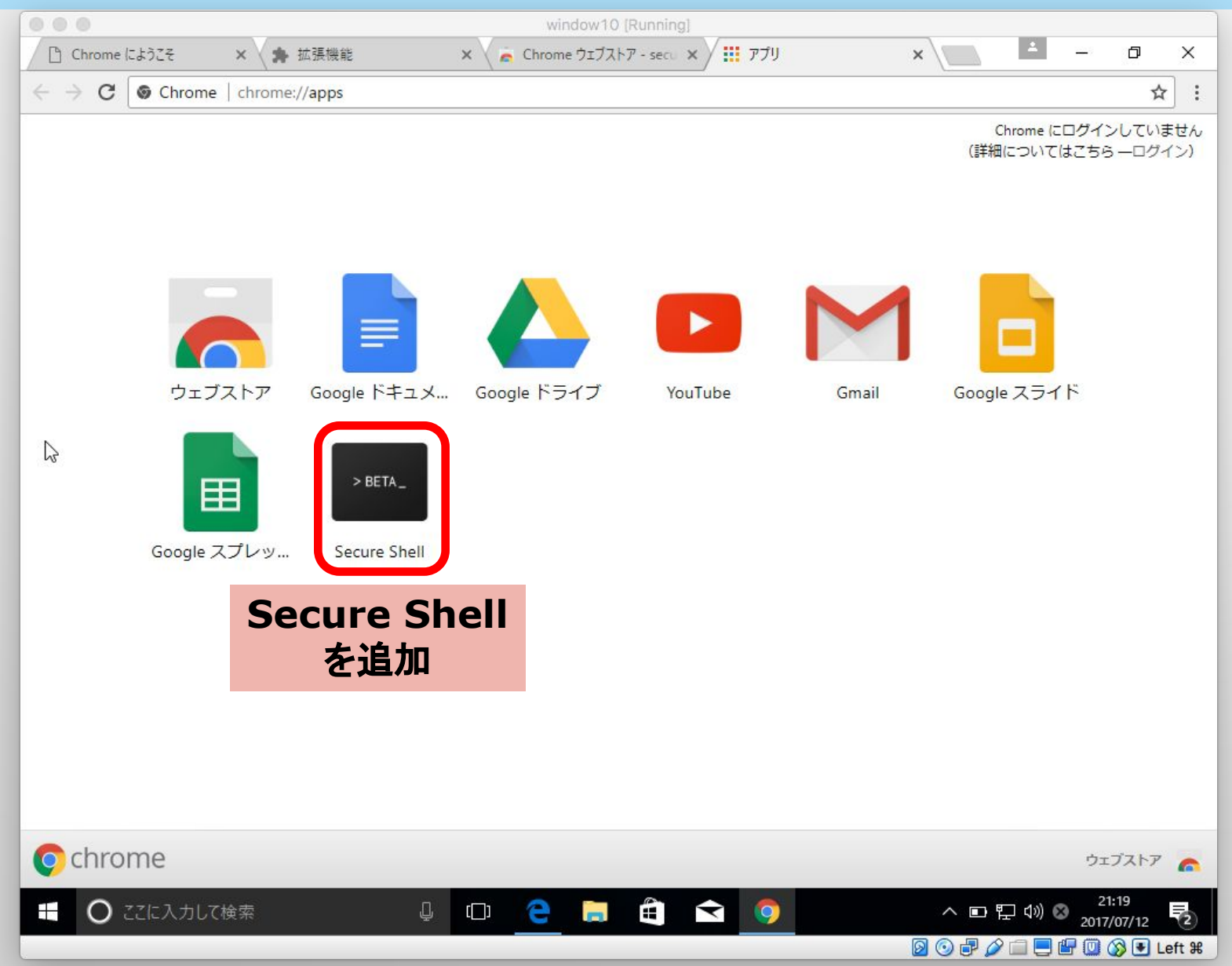
ssh設定


window10 [Running]
Chrome ようこそ
拡張機能
Chrome ウェブストア - 拡張
保護された通信 | https://chrome.google.com/webstore/category/extensions?hl=ja
Chrome ウェブストア
ログイン
secure shell
secure shell
secure shell 1366x768
secure shell 1920x1080 triangle
secure shell 1280x1024
secure shell 1280x1024 triangle
secure shell 1600x900
secure shell 1920x1080
Secure Shell を検索
DocuSign + Gmail
Securely send electronic signatures from Gmail and Drive
カテゴリ
すべて
機能
オフラインで実行
By Google
無料
Android で利用可能
Google ドライブに対応
評価
以上
以上
基本のアプリ
Chrome を最大限に活用するために揃えておきたい拡張機能
すべてを表示
Office
Office Online
Hangouts
Google ハングアウト
Google Keep
Google Keep Chrome 拡張機能
ここに入力して検索
21:18
2017/07/12
Left ⌘

ssh設定

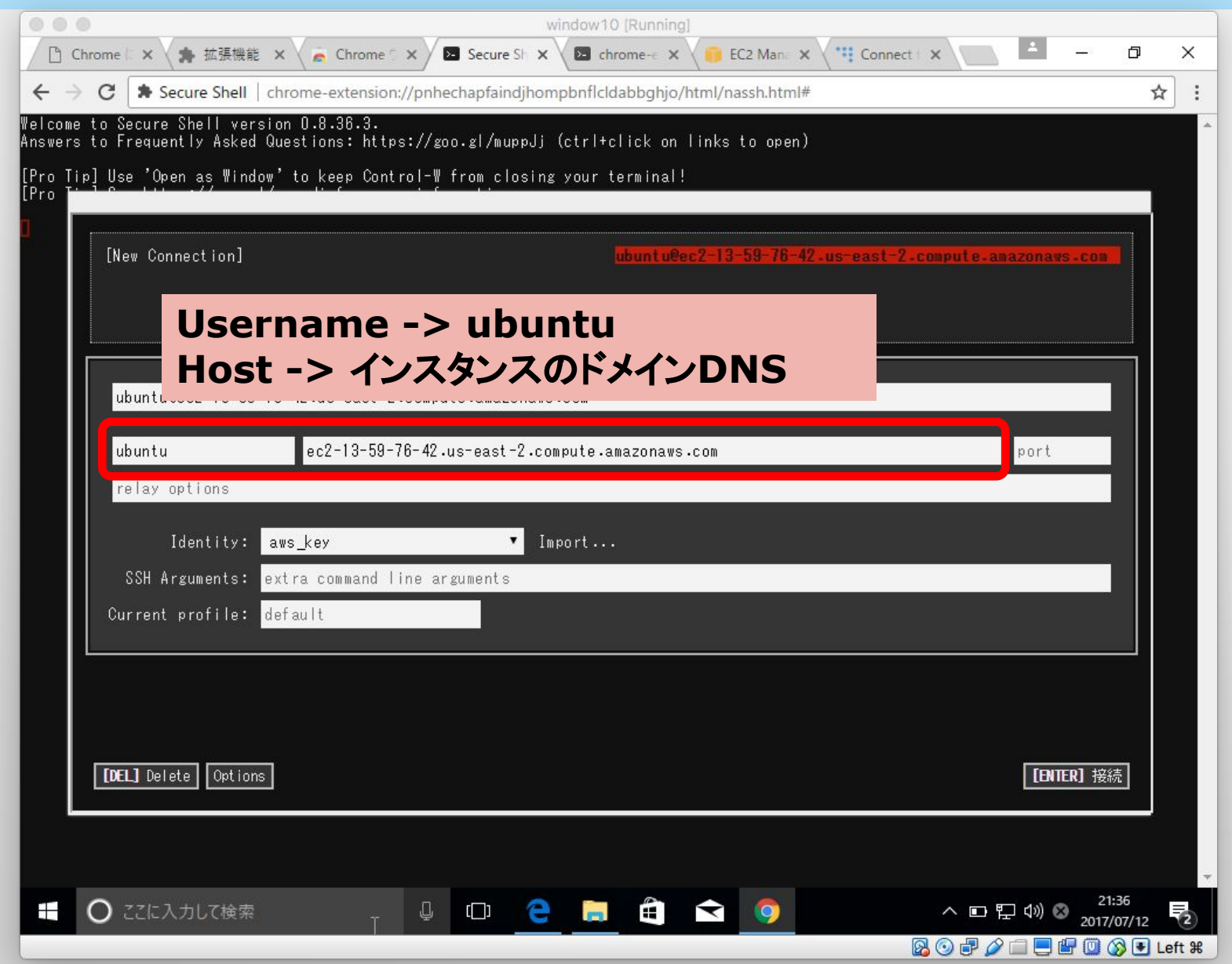

window10 [Running]
Chrome ようこそ
拡張機能
Chrome ウェブストア - secu
保護された通信 | https://chrome.google.com/webstore/search/secure%20shell?hl=ja
オフラインで実行
By Google
無料
Android で利用可能
Google ドライブに対応
評価
以上
適用于宙斯盾快速建立各个环境的SSH链接，该插件依赖于Secure Shell.
仕事効率化
(4)
Keep Awake
extensions@chromium.org
Override system power-saving settings.
CHROME に追加
仕事効率化
(403)
アプリ
Secure Shellを
CHROMEに追加
Secure Shell
chrome-secure-shell-publishers
Terminal emulator and SSH client.
CHROME に追加
(2955)
Crosh Window
Chromium OS Developers
*** For Chrome OS only *** You must have Secure Shell installed for this to work:...
CHROME に追加
デベロッパー ツール
(140)
狼アベンジャー
online.droidtom.com 提供
ダッシュ、かむと敵を殺す！狼男が復讐をするため、今その時間！
CHROME に追加
アーケード、アクション
(21)
ここに入力して検索
21:18
スピーカー: 67%
Left ⌘

ssh設定

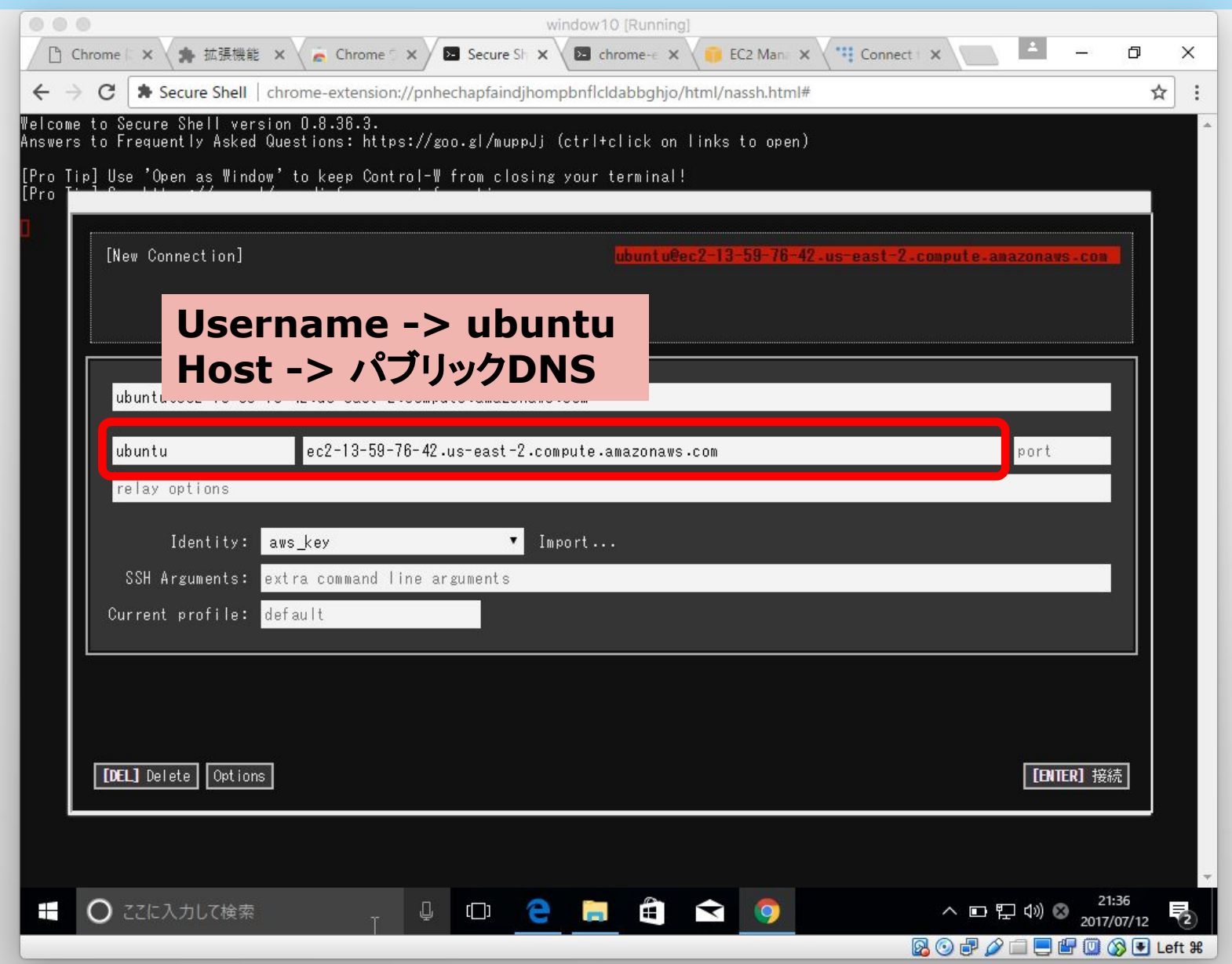


Secure Shell を追加

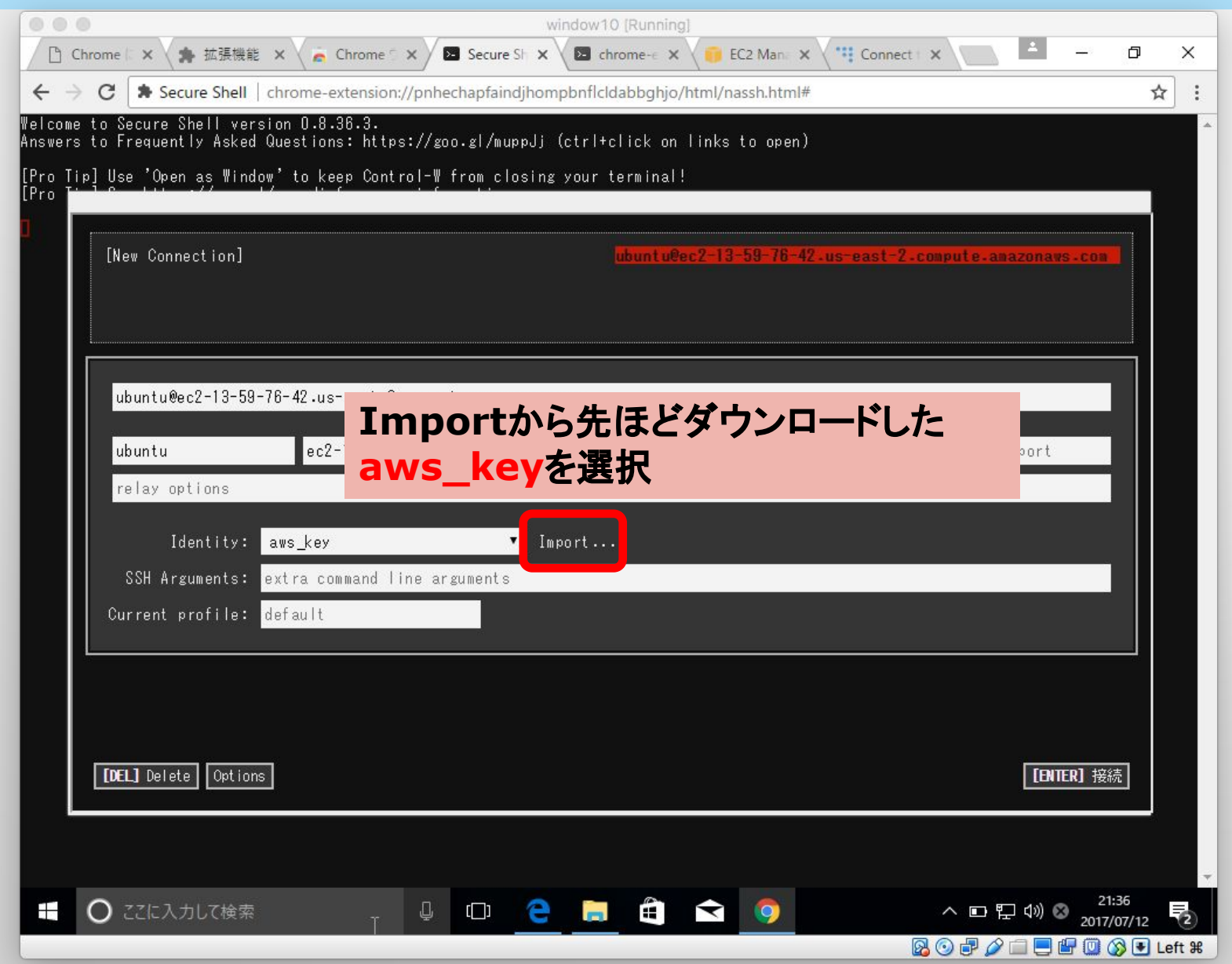
ssh設定


window10 [Running]
Secure Shell
chrome-extension://pnhechapfaindjhompbnflcldabbghjo/html/nassh.html#
Welcome to Secure Shell version 0.8.36.3.
Answers to Frequently Asked Questions: https://goo.gl/muppJj (ctrl+click on links to open)
[Pro Tip] Use 'Open as Window' to keep Control-W from closing your terminal!
[New Connection]
ubuntu@ec2-13-59-76-42.us-east-2.compute.amazonaws.com
Username -> ubuntu
Host -> インスタンスのドメインDNS
ubuntu
ec2-13-59-76-42.us-east-2.compute.amazonaws.com
port
relay options
Identity:
aws_key
Import...
SSH Arguments:
extra command line arguments
Current profile:
default
[DEL] Delete
Options
[ENTER] 接続
ここに入力して検索
21:36
2017/07/12
Left ⌘

ssh設定

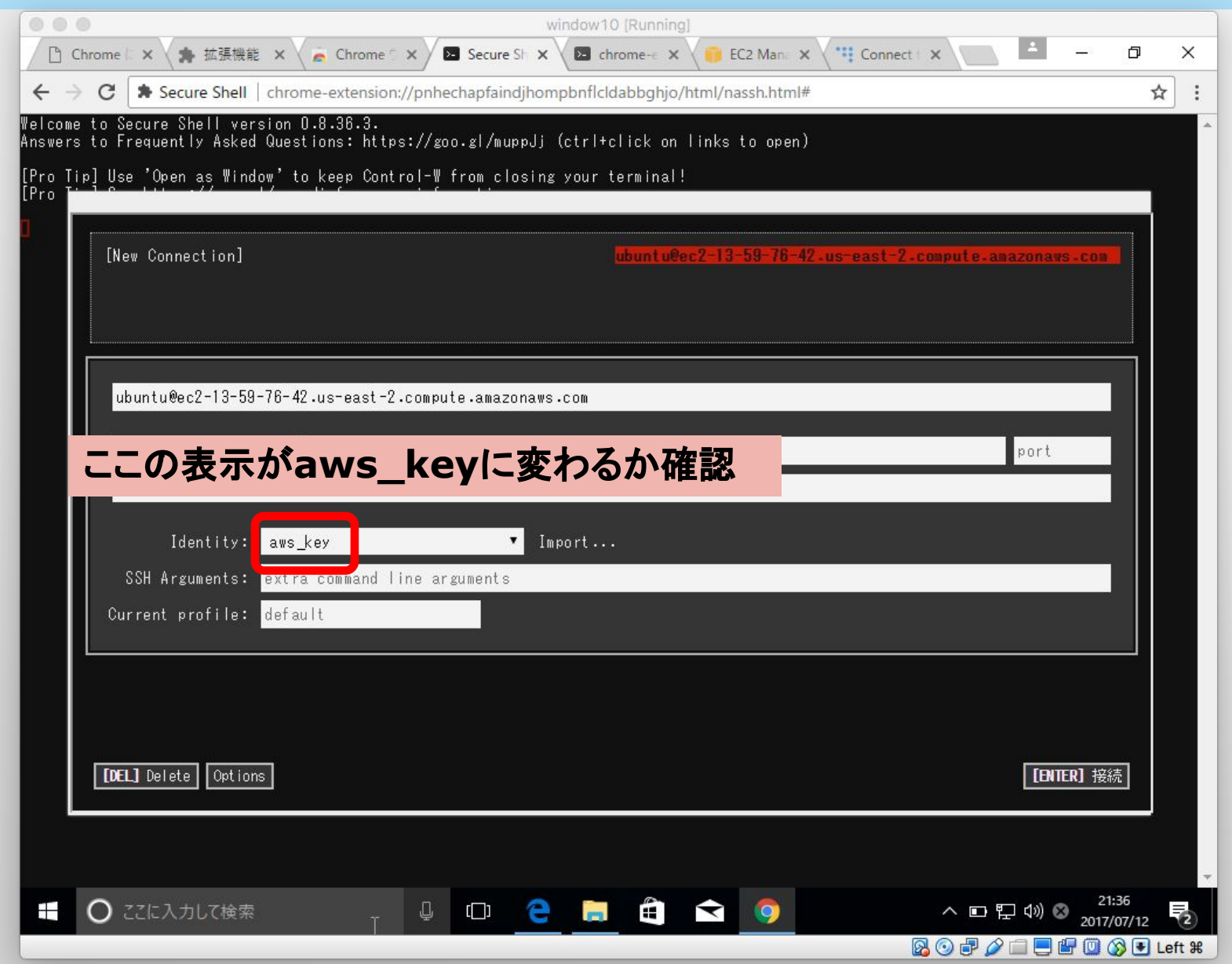

window10 [Running]
Secure Shell
chrome-extension://pnhechapfaindjhompbnflcldabbghjo/html/nassh.html#
Welcome to Secure Shell version 0.8.36.3.
Answers to Frequently Asked Questions: https://goo.gl/muppJj (ctrl+click on links to open)
[Pro Tip] Use 'Open as Window' to keep Control-W from closing your terminal!
[New Connection]
ubuntu@ec2-13-59-76-42.us-east-2.compute.amazonaws.com
Username -> ubuntu
Host -> パブリックDNS
ubuntu
ec2-13-59-76-42.us-east-2.compute.amazonaws.com
port
relay options
Identity:
aws_key
Import...
SSH Arguments:
extra command line arguments
Current profile:
default
[DEL] Delete
Options
[ENTER] 接続
ここに入力して検索
21:36
2017/07/12
Left ⌘

ssh設定



Importから先ほどダウンロードした aws_keyを選択

ssh設定


window10 [Running]
Secure Shell
chrome-extension://pnhechapfaindjhompbnflcldabbghjo/html/nassh.html#
Welcome to Secure Shell version 0.8.36.3.
Answers to Frequently Asked Questions: https://goo.gl/muppJj (ctrl+click on links to open)
[Pro Tip] Use 'Open as Window' to keep Control-W from closing your terminal!
[New Connection]
ubuntu@ec2-13-59-76-42.us-east-2.compute.amazonaws.com
ubuntu@ec2-13-59-76-42.us-east-2.compute.amazonaws.com
port
ここの表示がaws_keyに変わるか確認
Identity:
aws_key
Import...
SSH Arguments:
extra command line arguments
Current profile:
default
[DEL] Delete
Options
[ENTER] 接続
ここに入力して検索
21:36
2017/07/12
Left ⌘

ssh設定

window10 [Running]

Secure Shell | chrome-extension://pnhechapfaindjhompbnflcldabbghjo/html/nassh.html#

```
Welcome to Secure Shell version 0.8.36.3.
Answers to Frequently Asked Questions: https://goo.gl/muppJj (ctrl+click on links to open)

[Pro Tip] Use 'Open as Window' to keep Control-W from closing your terminal!
```

[New Connection] ubuntu@ec2-13-59-76-42.us-east-2.compute.amazonaws.com

ubuntu@ec2-13-59-76-42.us-east-2.compute.amazonaws.com

ubuntu | ec2-13-59-76-42.us-east-2.compute.amazonaws.com | port

relay options

Identity: aws_key Import...

SSH Arguments: extra command line arguments

Current profile: default

Enter

[DEL] Delete | Options | [ENTER] 接続

ここに入力して検索 21:36 2017/07/12 Left ⌘

Hands on List

1. 環境設定
   - インスタンス作成
   - ssh設定
   - 環境構築
   - BitBucket設定
   - Jupyter Notebook設定
2. サンプルプログラムの実行
3. コード解説
4. テスト

環境構築

- Python環境を順番に構築していきます
- コマンドをコンソール上で実行してください
- 記載コマンド先頭の“$”は入力不要です
- **同じコマンドを複数回実行すると不具合が生じる場合があります**
  - うまく動かなかったらファシリテーターをお呼びください

環境構築(パッケージ更新)

```
$sudo apt-get -y update;sudo apt-get –y\
upgrade
```

(“$” は入力の必要はありません)

環境構築(pyenv)

```
$ git clone git://github.com/yyuu/pyenv.git ~/.pyenv
$ echo 'export PYENV_ROOT="$HOME/.pyenv"' >> ~/.bashrc
$ echo 'export PATH="$PYENV_ROOT/bin:$PATH"' >> ~/.bashrc
$ echo 'eval "$(pyenv init -)"' >> ~/.bashrc
$ source ~/.bashrc
```

(“$” は入力の必要はありません)

※echo から始まるコマンドの入力を間違えた場合、修正する必要があるのでファシリテーターをお呼びください。

環境構築(anaconda)

```
$ pyenv install anaconda3-4.4.0
$ pyenv global anaconda3-4.4.0
$ echo 'export \
PATH="$PYENV_ROOT/versions/anaconda3-\
4.4.0/bin:$PATH"' >> ~/.bashrc
$ source ~/.bashrc
```

(“$” は入力の必要はありません)

※echo から始まるコマンドの入力を間違えた場合、修正する必要があるのでファシリテーターをお呼びください。

環境構築(Python module)

```
$conda install -y tensorflow
$conda install -y keras
$conda install -y pillow
```

(“$” は入力の必要はありません)

Hands on List

1. 環境設定
   - インスタンス作成
   - ssh設定
   - 環境構築
   - BitBucket設定
   - Jupyter Notebook設定
2. サンプルプログラムの実行
3. コード解説
4. テスト

サンプルコード取得

```
$git clone \
https://bitbucket.org/teamgoods/ch01.git
```

(“$” は入力の必要はありません)

Hands on List

1. 環境設定
   - インスタンス作成
   - ssh設定
   - 環境構築
   - BitBucket設定
   - Jupyter Notebook設定
2. サンプルプログラムの実行
3. コード解説
4. テスト

Jupyter Notebookサーバ接続

```
$jupyter notebook --generate-config
$echo "c.NotebookApp.ip ='*'" >>\
~/.jupyter/jupyter_notebook_config.py
$echo "c.NotebookApp.port = 8888" >>\
~/.jupyter/jupyter_notebook_config.py
```

(“$” は入力の必要はありません)

※echo から始まるコマンドの入力を間違えた場合、修正する必要があるのでファシリテーターをお呼びください。

Jupyter Notebookサーバ接続

```
$jupyter-notebook
```
で表示されたURLへアクセス

(“$” は入力の必要はありません)

Jupyter Notebookサーバ接続

window10 [Running]

ubuntu@ip-172-31-18-9

Secure Shell | chrome-extension://pnhechapfaindjhompbnflcldabbghjo/html/nassh.html#profile-id:ac1d

```
key_load_public: invalid format
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.4.0-1020-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  Get cloud support with Ubuntu Advantage Cloud Guest:
    http://www.ubuntu.com/business/services/cloud

22 packages can be updated.
0 updates are security updates.


*** System restart required ***
Last login: Wed Jul 12 12:59:13 2017 from 203.178.135.47
ubuntu@ip-172-31-18-94:~$ ls
ch01  pyenv-pip-rehash
ubuntu@ip-172-31-18-94:~$ cd ch01
ubuntu@ip-172-31-18-94:~/ch01$ jupyternotebook
jupyternotebook: command not found
ubuntu@ip-172-31-18-94:~/ch01$ jupyter-notebook
[I 07:10:21.507 Not                                  to /run/user/1000/jupyter/notebook_cookie_secret
[W 07:10:21.545 Not                                  ning on all IP addresses and not using encryption. This is not
mended.
[I 07:10:21.557 Not                                  y: /home/ubuntu/ch01
[I 07:10:21.557 Not
[I 07:10:21.558 Not                                  ttp://[all ip addresses on your system]:8888/?token=eca6bfbece6b
5461f6cb402126ed07c
[I 07:10:21.558 Not                                  shut down all kernels (twice to skip confirmation).
[W 07:10:21.558 Not                                  e runnable browser.
[C 07:10:21.558 Not

    Copy/paste this URL into your browser when you connect for the first time,
    to login with a token:
        http://localhost:8888/?token=eca6bfbece6b39ed5461f6cb402126ed07c2fb35c6209813
```

localhost
を，インスタンスの
”パブリックDNS”
に書き換えてアクセス

ここに入力して検索

16:10
2017/07/13

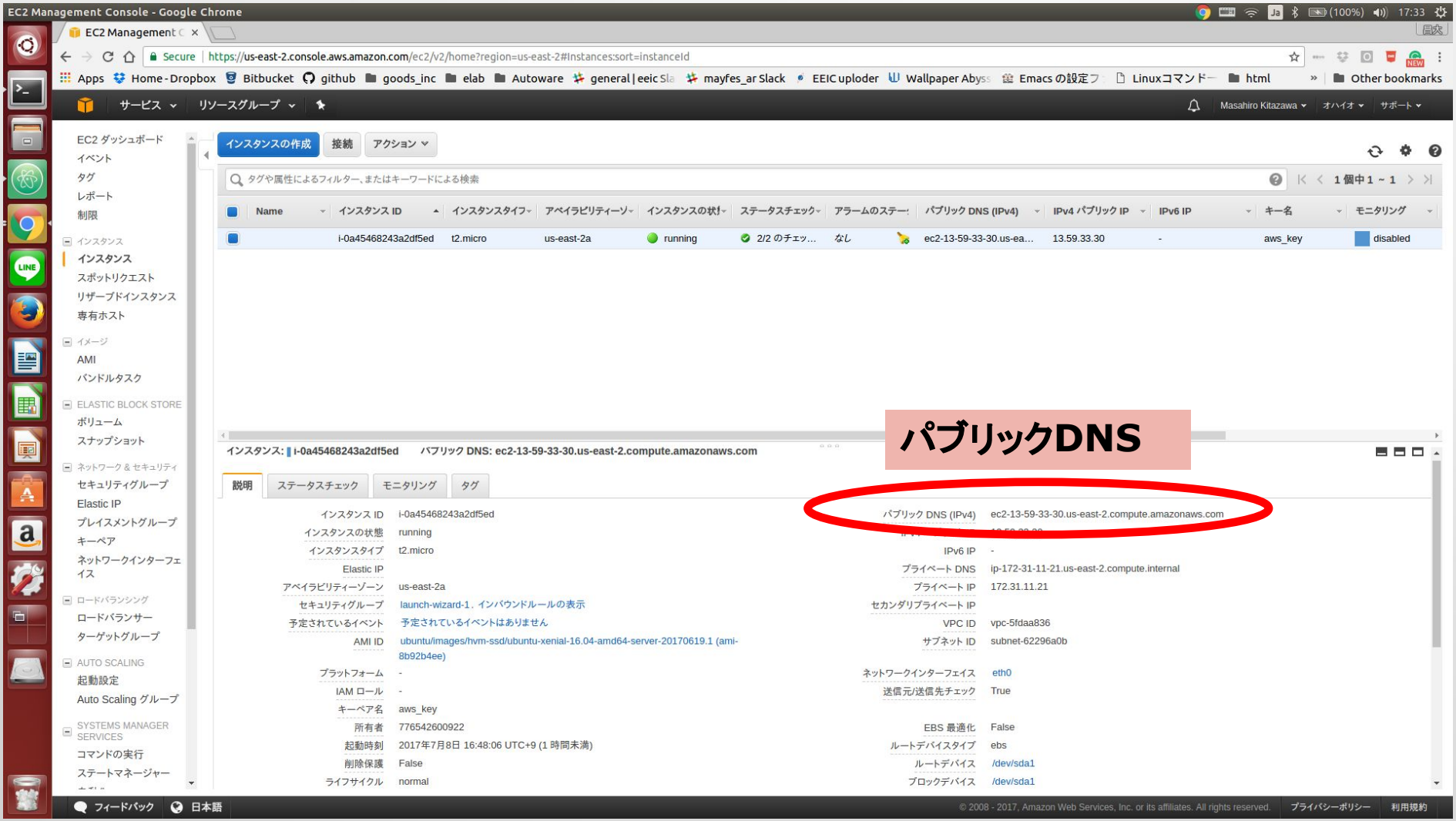
Left ⌘

Jupyter Notebookサーバ接続

- http://localhost:8888/?token=xxxxxx...
- http://ec2-xxxx.amazonaws.com:8888/?token=xxxxxx...

localhost
を
”パブリックDNS”
に変更

Jupyter Notebookサーバ接続



Hands on List

1. 環境設定
   - インスタンス作成
   - ssh設定
   - 環境構築
   - BitBucket設定
   - Jupyter Notebook設定
2. サンプルプログラムの実行
3. コード解説
4. テスト

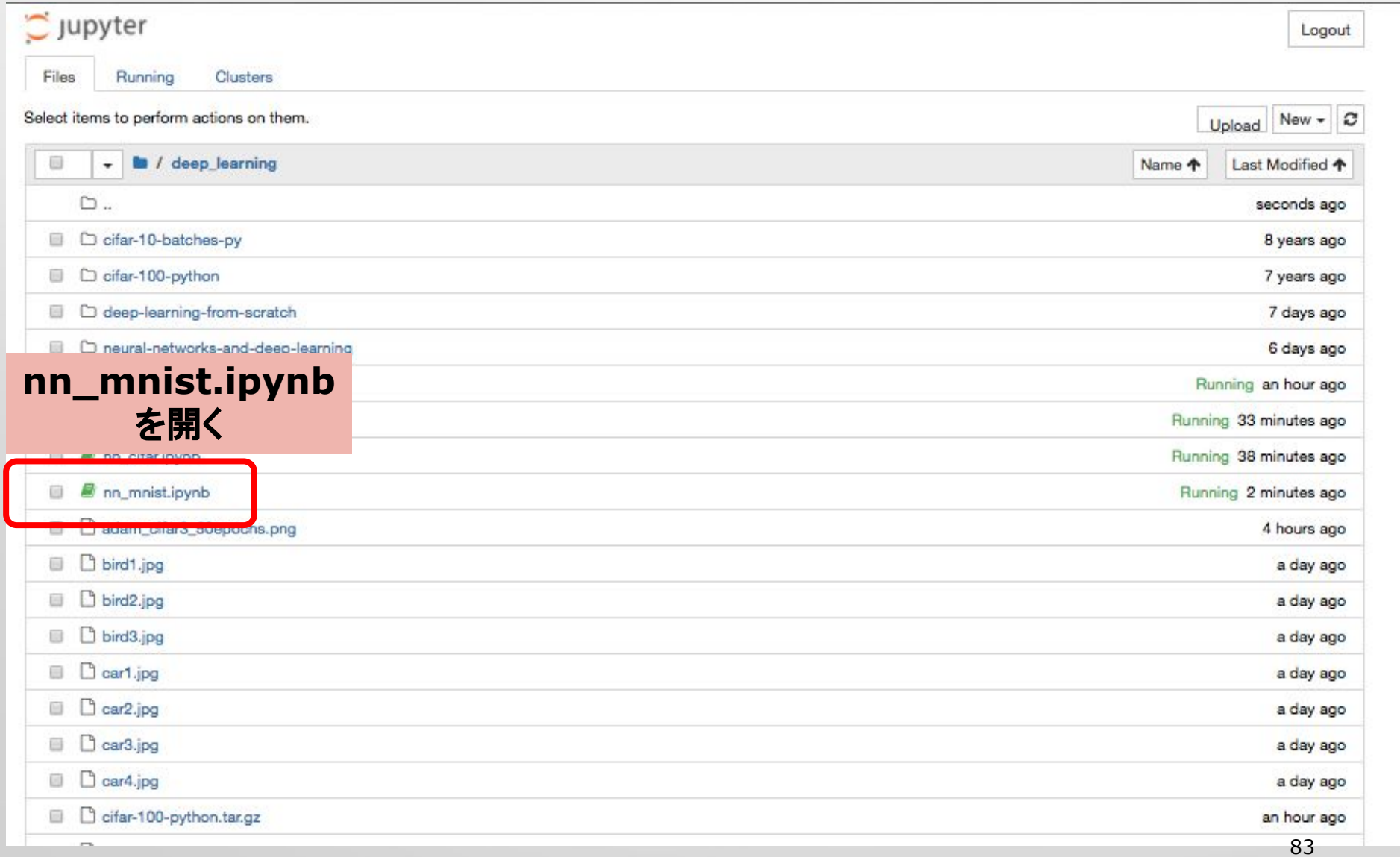
サンプルコードの実行

- Jupyter Notebookの使い方・各種コマンド
- https://algorithm.joho.info/programming/jupyter-notebook-windows-linux/

ファイル一覧

アクセスすると
ファイル一覧画面が表示される

ファイル一覧



nn_mnist.ipynb を開く

サンプルコードの実行

jupyter nn_mnist Last Checkpoint: Last Tuesday at 6:24 PM (autosaved)

Logout

File Edit View Insert Cell Kernel Help Trusted Python 3

Markdown

Kerasによるニューラルネット実装例(MNIST)

In [1]:
```
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout
from keras.optimizers import RMSprop
import h5py
```

```
Using TensorFlow backend.
```

In [71]:
```
# 自分で設定するパラメータ

batch_size = 1000
epochs = 10
```

In [72]:
```
# データ抽出
(x_train, y_train), (x_test, y_test) = mnist.load_data()
num_classes = 10
x_train = x_train.reshape(60000, 784)
x_test = x_test.reshape(10000, 784)
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')

# one-hot-label に変換
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)
```

```
60000 train samples
10000 test samples
```

サンプルコードの実行

jupyter nn_mnist Last Checkpoint: Last Tuesday at 6:24 PM (autosaved)

Logout

File Edit View Insert Cell Kernel Help Trusted Python 3

この箱一つ一つがセル。
セルごとにプログラミング・実行ができる
実行はセルを選択して"Shift+Enter"

Kerasによる

In [1]:
```
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout
from keras.optimizers import RMSprop
import h5py
```

```
Using TensorFlow backend.
```

In [71]:
```
# 自分で設定するパラメータ

batch_size = 1000
epochs = 10
```

In [72]:
```
# データ抽出
(x_train, y_train), (x_test, y_test) = mnist.load_data()
num_classes = 10
x_train = x_train.reshape(60000, 784)
x_test = x_test.reshape(10000, 784)
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')

# one-hot-label に変換
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)
```

```
60000 train samples
10000 test samples
```

サンプルコードの実行

jupyter nn_mnist Last Checkpoint: a minute ago (unsaved changes)

Logout

File Edit View Insert Cell Kernel Help

Trusted | Python 3

Run Cells
Run Cells and Select Below
Run Cells and Insert Below
Run All
Run All Above
Run All Below
Cell Type
Current Outputs
All Output

Run All をクリック

In [14]:
```
import keras
from keras.da
from keras.mo
from keras.lay
from keras.op
```

In [15]:
```
# 自分で設定するパラメータ
batch_size = 128
num_classes = 10
epochs = 5
```

In [16]:
```
# データ抽出
(x_train, y_train), (x_test, y_test) = mnist.load_data()

x_train = x_train.reshape(60000, 784)
x_test = x_test.reshape(10000, 784)
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')

# one-hot-label に変換
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)
```

```
60000 train samples
10000 test samples
```

サンプルコードの実行

jupyter nn_mnist Last Checkpoint: 31 minutes ago (unsaved changes)

Logout

File Edit View Insert Cell Kernel Help Trusted Python 3

Code

```
# スコアの計算・表示
score = model.evaluate(x_test, y_test, verbose=1)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_4 (Dense)              (None, 512)               401920
_________________________________________________________________
dropout_3 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_5 (Dense)              (None, 512)               262656
_________________________________________________________________
dropout_4 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_6 (Dense)              (None, 10)                5130
=================================================================
Total params: 669,706
Trainable params: 669,706
Non-trainable params: 0
_________________________________________________________________
Train on 60000 samples, validate on 10000 samples
Epoch 1/10
45184/60000 [==================>.......] - ETA: 2s - loss: 0.2811 - acc: 0.9144
```

学習の様子

In [5]:

```
import h5py
print(h5py)
# 学習モデルの保存
model.save("trained_data.h5")
```

```
<module 'h5py' from '/Users/kazuki/.pyenv/versions/3.6.1/lib/python3.6/site-packages/h5py/__init__.py'>
```

サンプルコードの実行

jupyter nn_mnist Last Checkpoint: 28 minutes ago (unsaved changes)

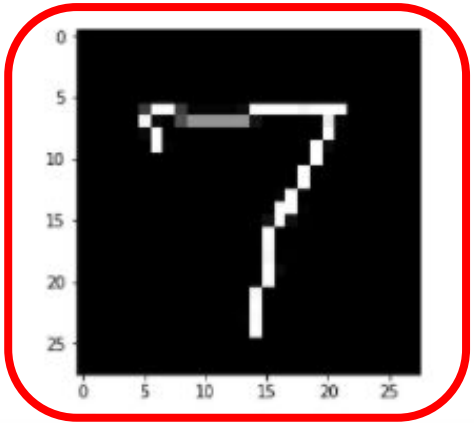
Logout

File Edit View Insert Cell Kernel Help Trusted Python 3

Code

In [60]:
```
#入力画像の表示
plt.imshow(im_list,cmap=plt.get_cmap('gray'))
```

Out[60]: <matplotlib.image.AxesImage at 0x12cd513c8>



テスト用画像が表示される

In [63]:
```
# 画像判別
test = im_list.reshape(1,784)
print(model.predict(test))
print('\n')
print('Result:',np.argmax(model.predict(test)[0]))
```

```
[[ 8.18891684e-04  2.97303940e-03  7.37489089e-02  2.03929767e-01
   9.53423791e-04  1.32523000e-03  4.42908859e-05  6.97649956e-01
   7.93705415e-03  1.06194271e-02]]
```

Result: 7

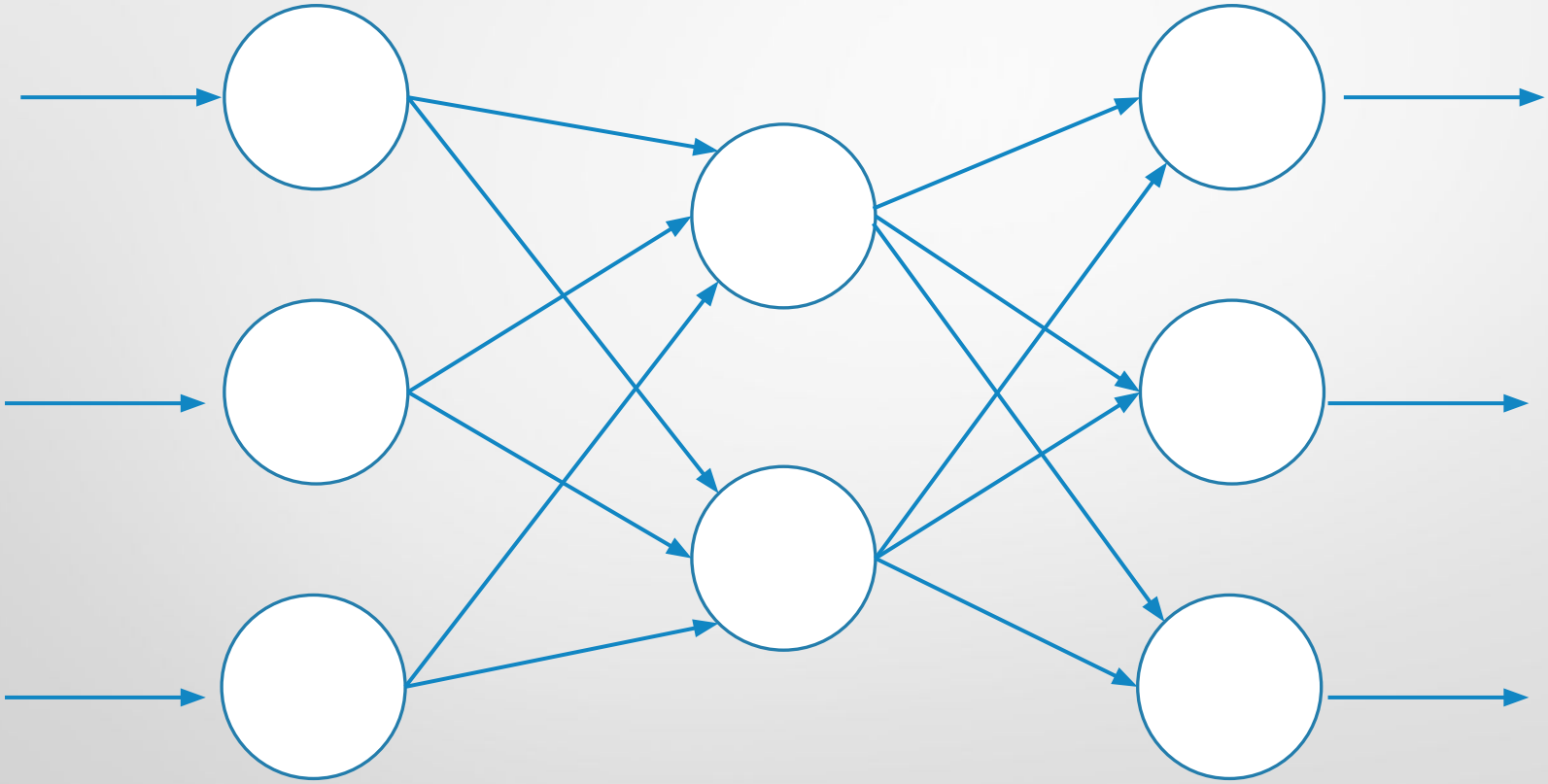
識別結果が表示される

Hands on List

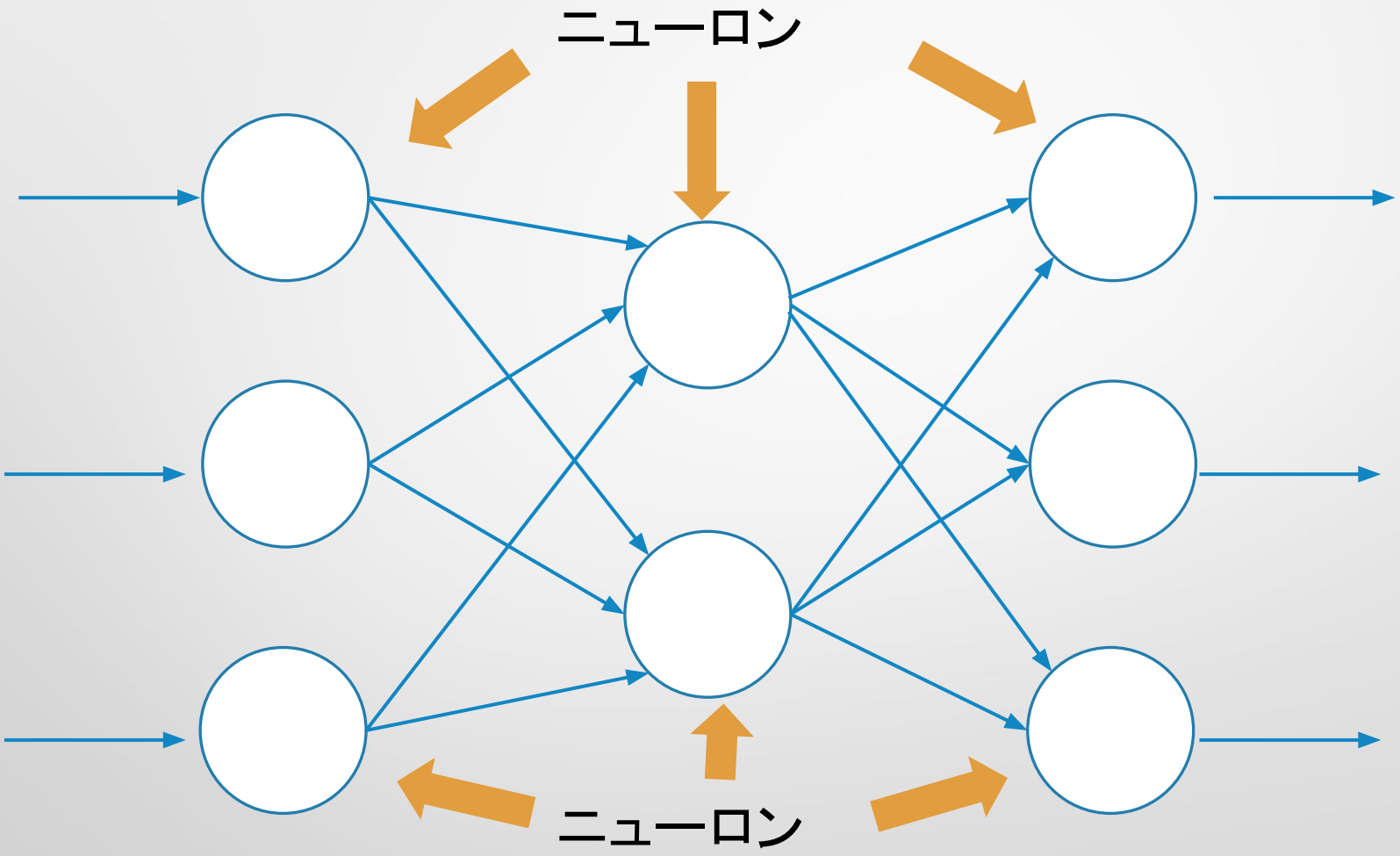
1. 環境設定
   - インスタンス作成
   - ssh設定
   - 環境構築
   - BitBucket設定
   - Jupyter Notebook設定
2. サンプルプログラムの実行
3. コード解説
4. テスト

モデル解説

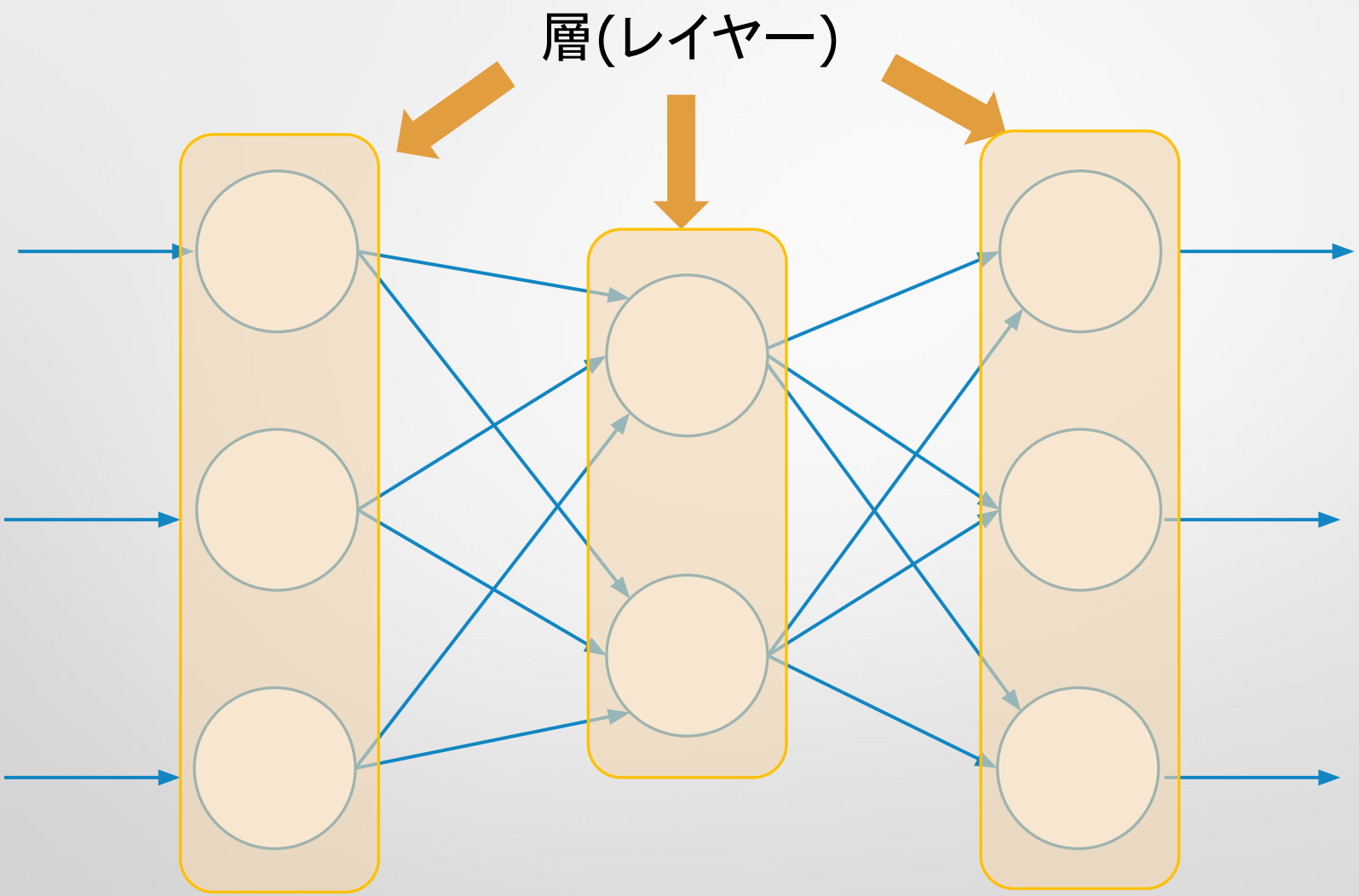
ニューラルネット(人間の脳のモデル)の構成は
人間が設定する



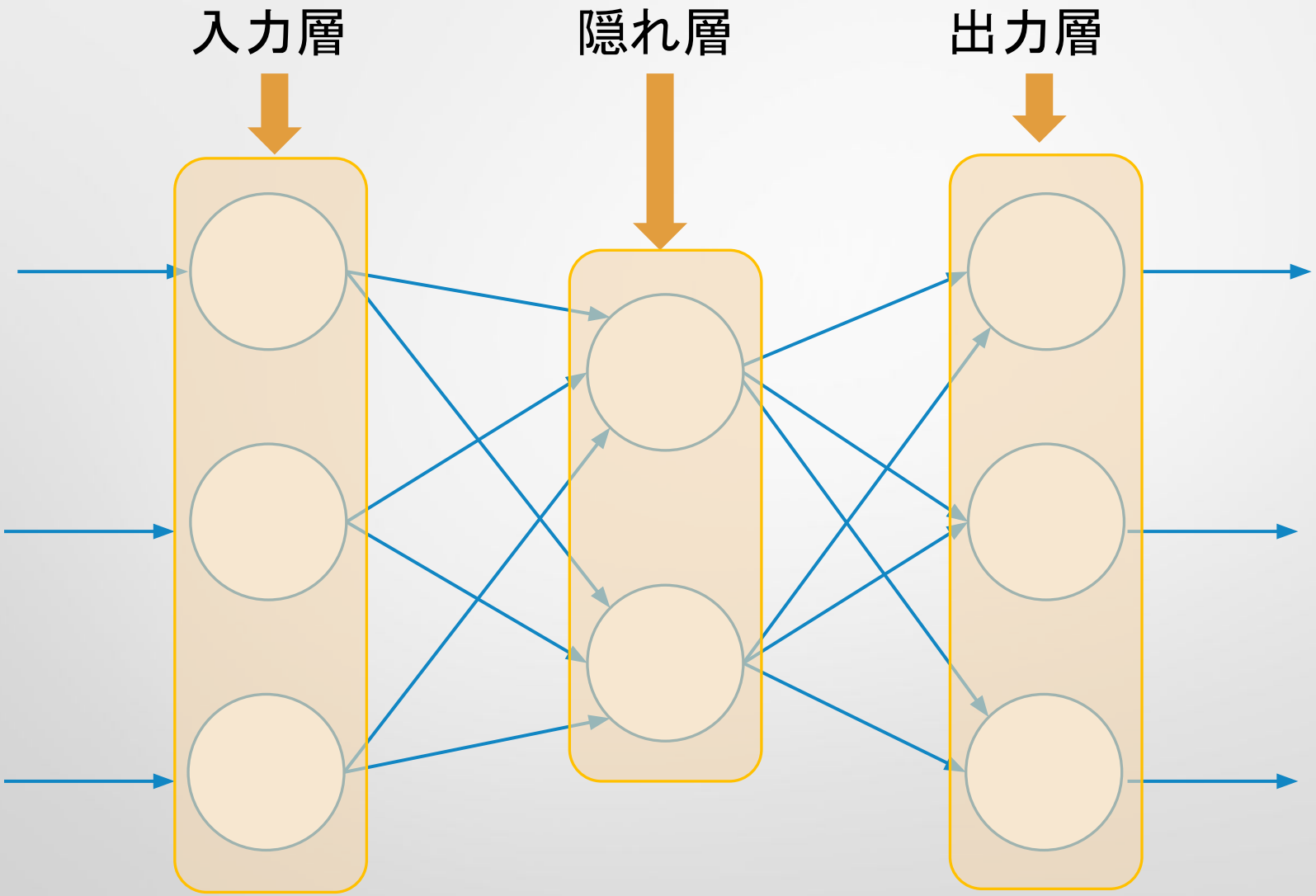
モデル解説



モデル解説



モデル解説

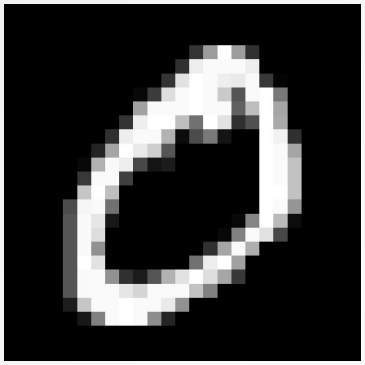


3層ニューラルネットワーク

画像の読み込み

ピクセルごとに，RGB値が割り当てられている

入力の形式

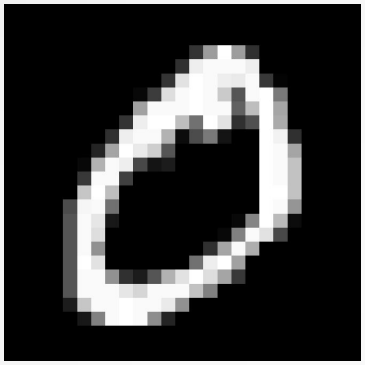


画素ごとのRGB値の表(行列)にする

$$\begin{bmatrix} [0,100,3] & \cdots & [10,10,3] \\ \vdots & \ddots & \vdots \\ [0,20,50] & \cdots & [2,0,0] \end{bmatrix}$$

$$[0, 100, 3, \cdots, 10,10,3, \cdots, 0,20,50, \cdots 2, 0, 0]$$

入力の形式

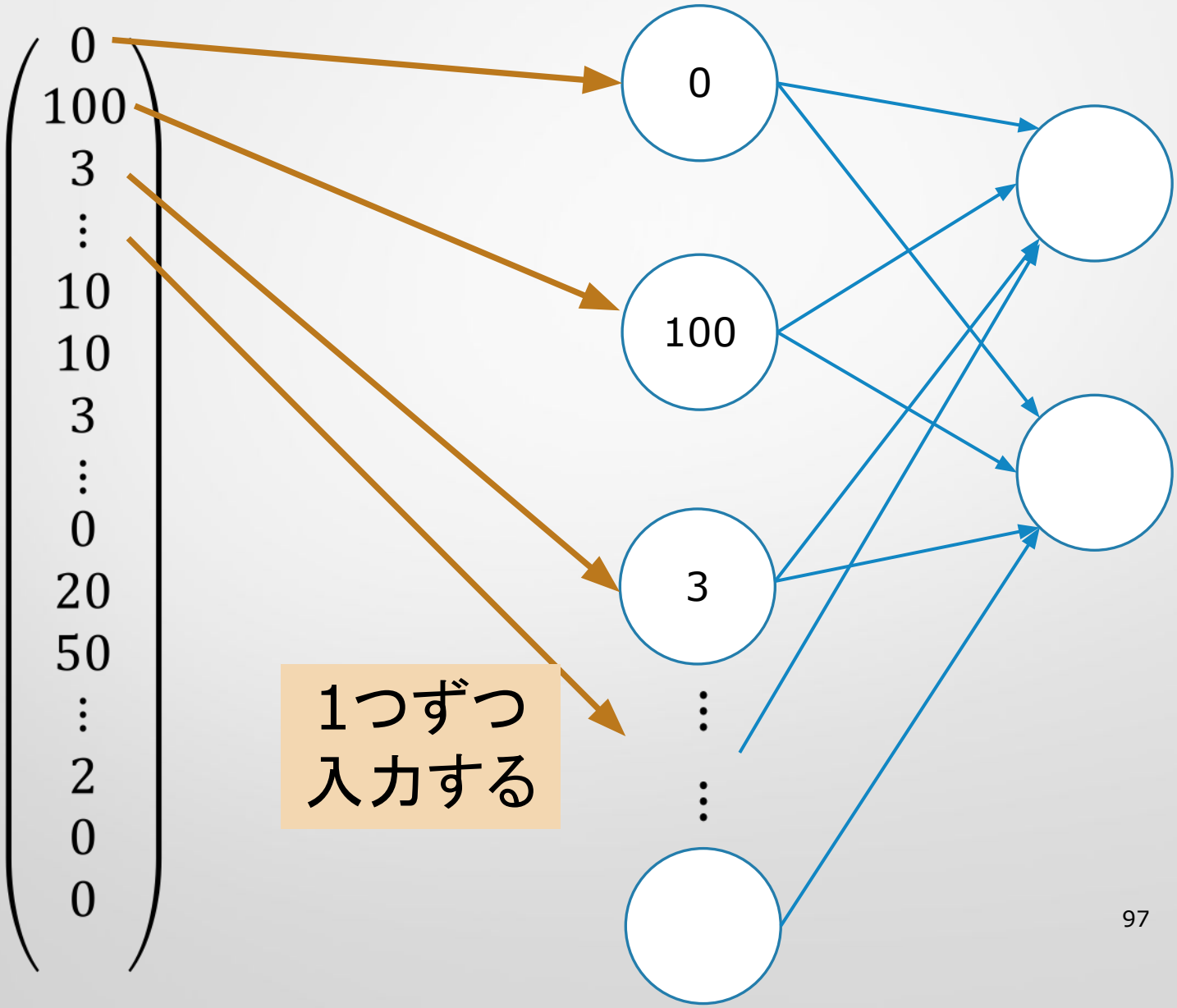


$$\begin{bmatrix} [0,100,3] & \cdots & [10,10,3] \\ \vdots & \ddots & \vdots \\ [0,20,50] & \cdots & [2,0,0] \end{bmatrix}$$

1行に並べ直す

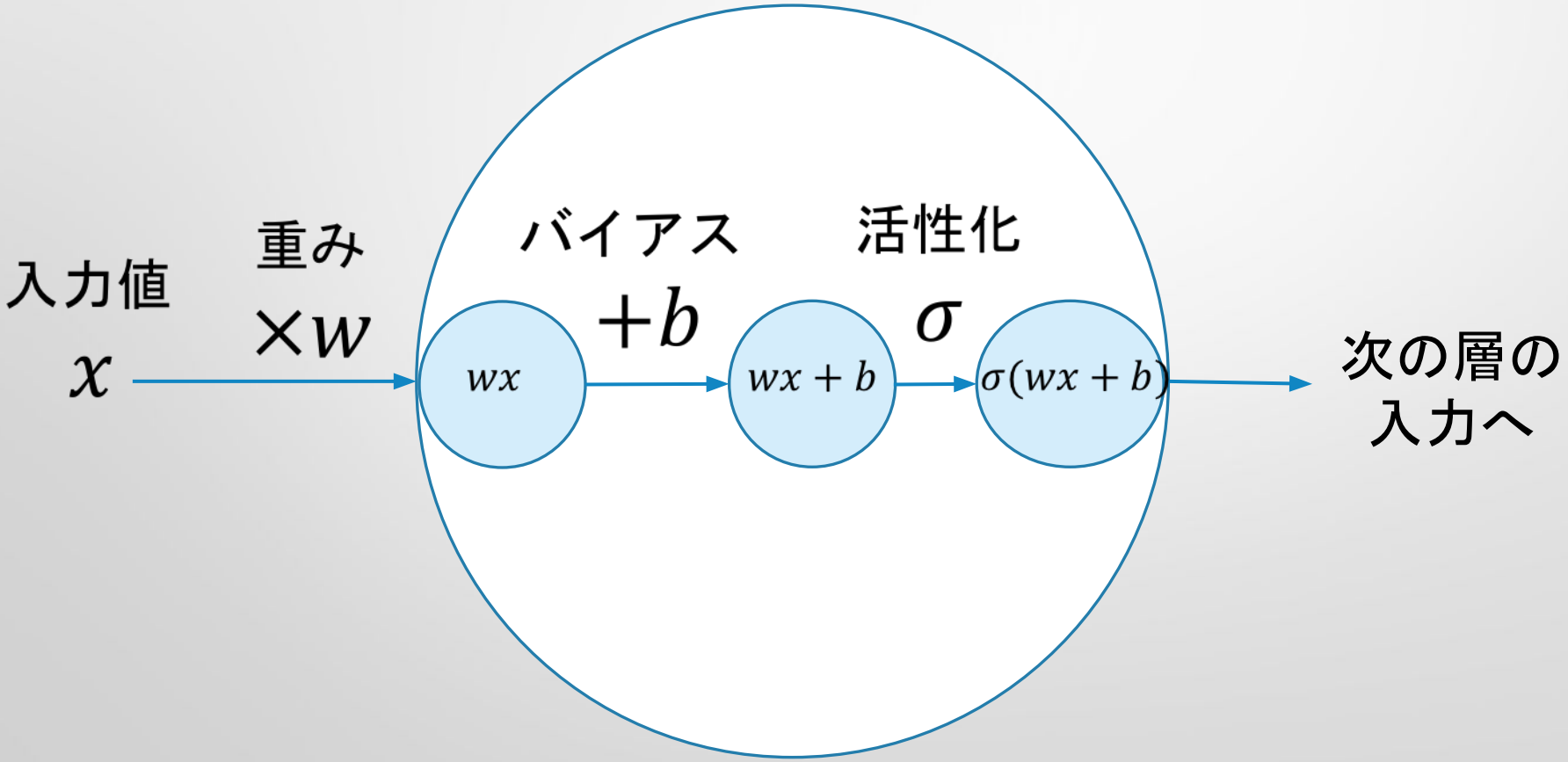
$$[0, 100, 3, \cdots, 10,10,3, \cdots, 0,20,50, \cdots 2, 0, 0]$$

入力の形式



ニューロンの仕組み

重み・バイアスを
ニューラルネットが自動で学習する



計算の流れ(数式で表すと)

入力値 $\mathbb{X}$　重み $\mathbb{W}$　バイアス $\mathbb{b}$

→ 次の層 $\sigma(\mathbb{W}\,\mathbb{X} + \mathbb{b})$

$$\begin{pmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{pmatrix} \begin{pmatrix} x_{11} & \cdots & x_{n1} \\ x_{12} & \ddots & \vdots \\ x_{13} & \cdots & x_{n3} \end{pmatrix} + \begin{pmatrix} b_{11} \\ b_{12} \end{pmatrix}$$

1枚の画像にあたる

計算の流れ(数式で表すと)

入力値 $\mathbb{X}$　重み $\mathbb{W}$　バイアス $\mathbb{b}$

→ 次の層 $\sigma(\mathbb{W}\,\mathbb{X} + \mathbb{b})$

$$\begin{pmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{pmatrix} \begin{pmatrix} x_{11} & \cdots & x_{n1} \\ x_{12} & \ddots & \vdots \\ x_{13} & \cdots & x_{n3} \end{pmatrix} + \begin{pmatrix} b_{11} \\ b_{12} \end{pmatrix}$$

たくさん並んでいる

計算の流れ(数式で表すと)

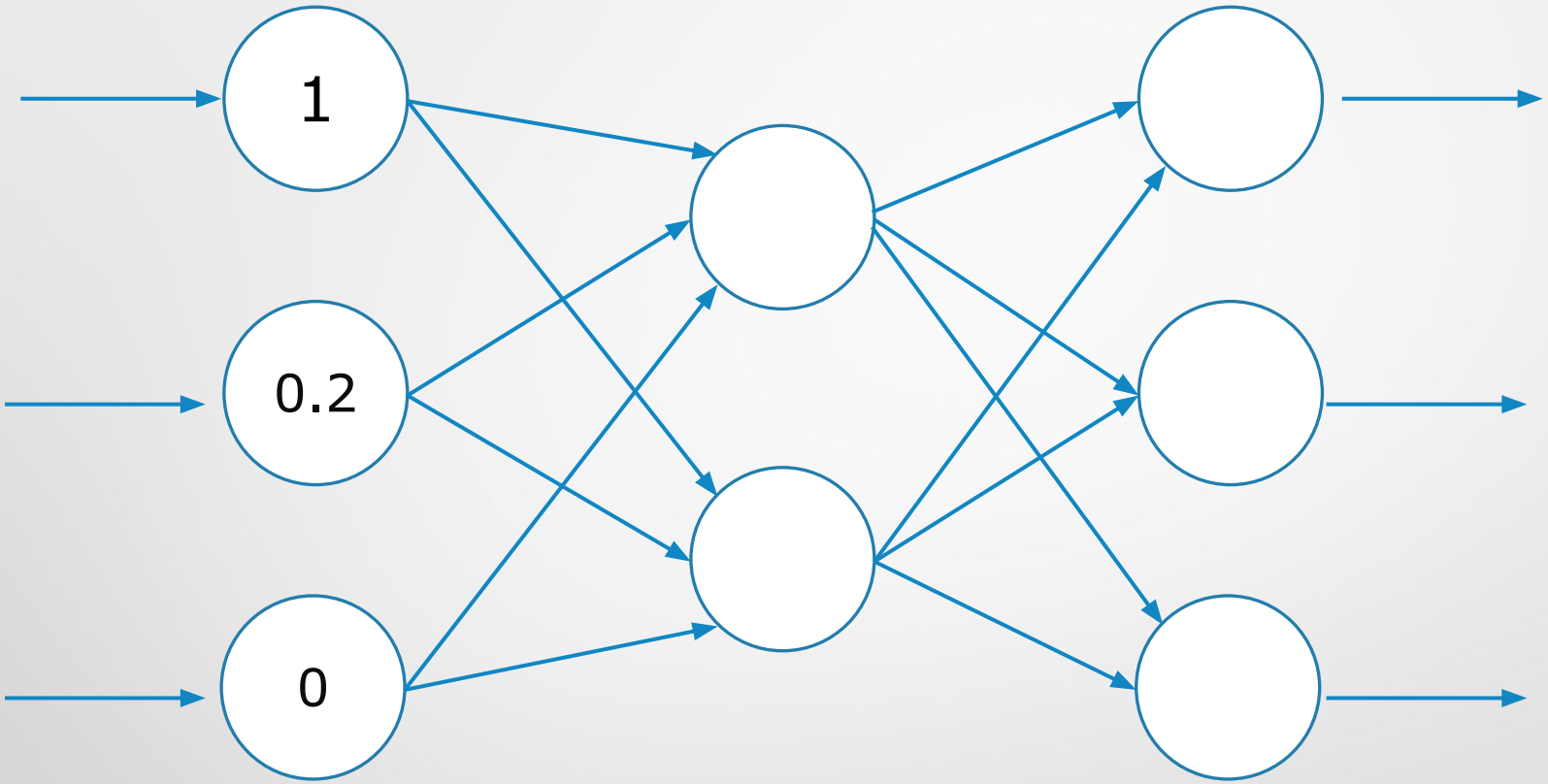
入力値 $\mathbb{X}$　重み $\mathbb{W}$　バイアス $\mathbb{b}$

→ 次の層 $\sigma(\mathbb{W}\,\mathbb{X} + \mathbb{b})$

$$\begin{pmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{pmatrix} \begin{pmatrix} x_{11} & \cdots & x_{n1} \\ x_{12} & \ddots & \vdots \\ x_{13} & \cdots & x_{n3} \end{pmatrix} + \begin{pmatrix} b_{11} \\ b_{12} \end{pmatrix}$$

並んでいる列数
＝バッチサイズ

計算の流れ(例)



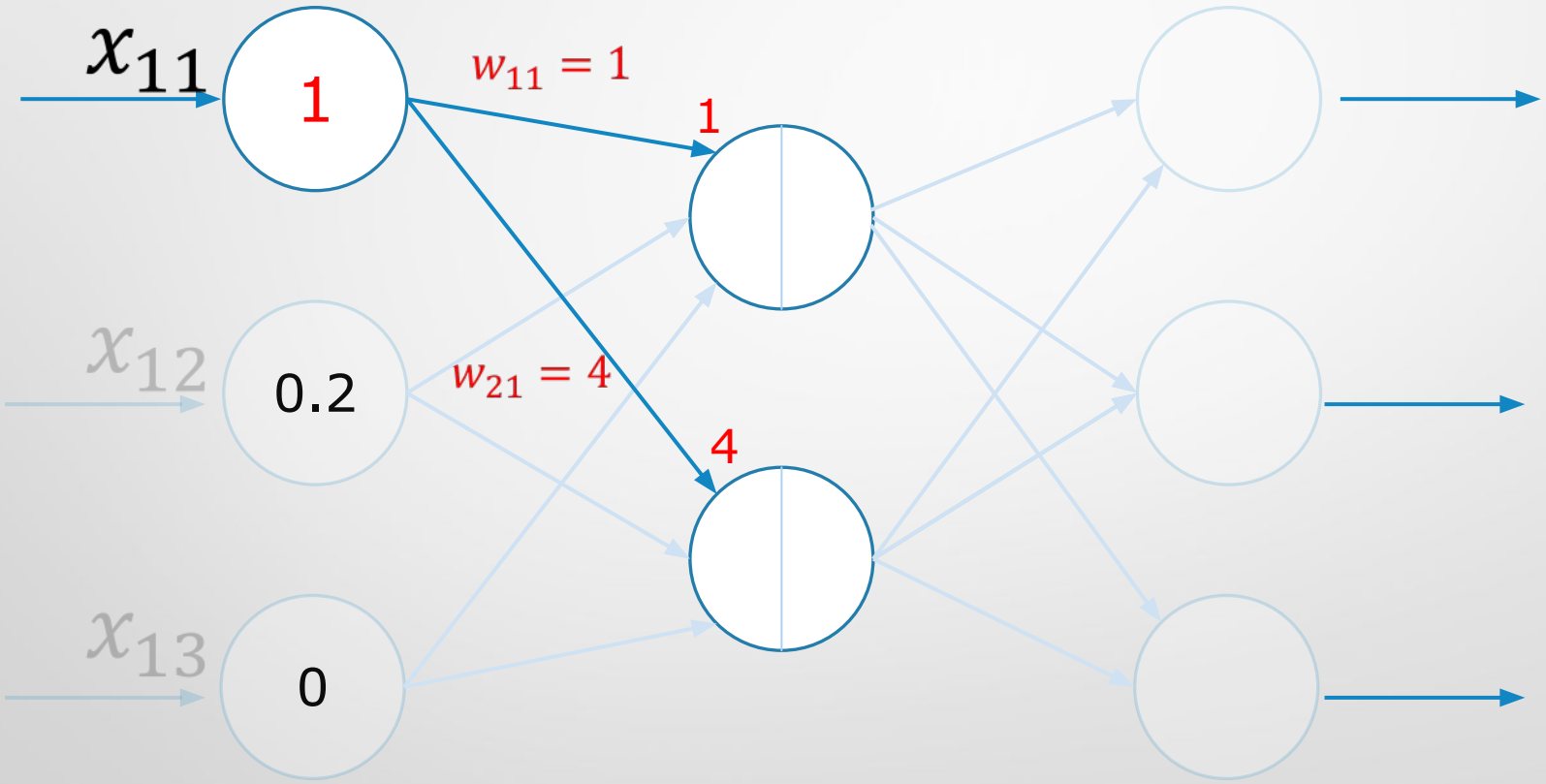
重み行列

$$\begin{pmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 5 \\ 4 & 3 & 3 \end{pmatrix}$$

バイアス

$$\begin{pmatrix} b_{11} \\ b_{12} \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

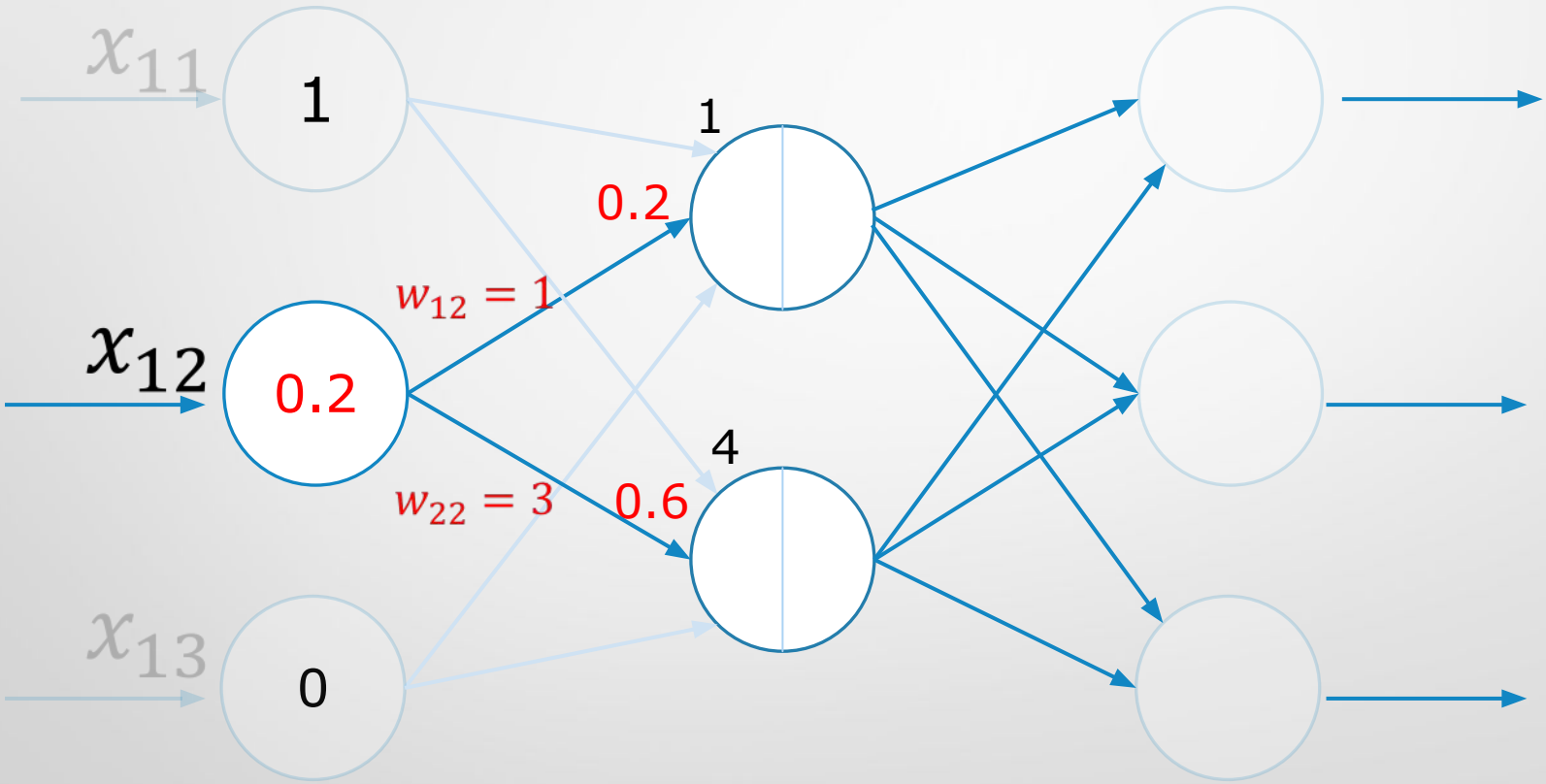
計算の流れ



重み行列

$$W = \begin{pmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 5 \\ 4 & 3 & 3 \end{pmatrix}$$

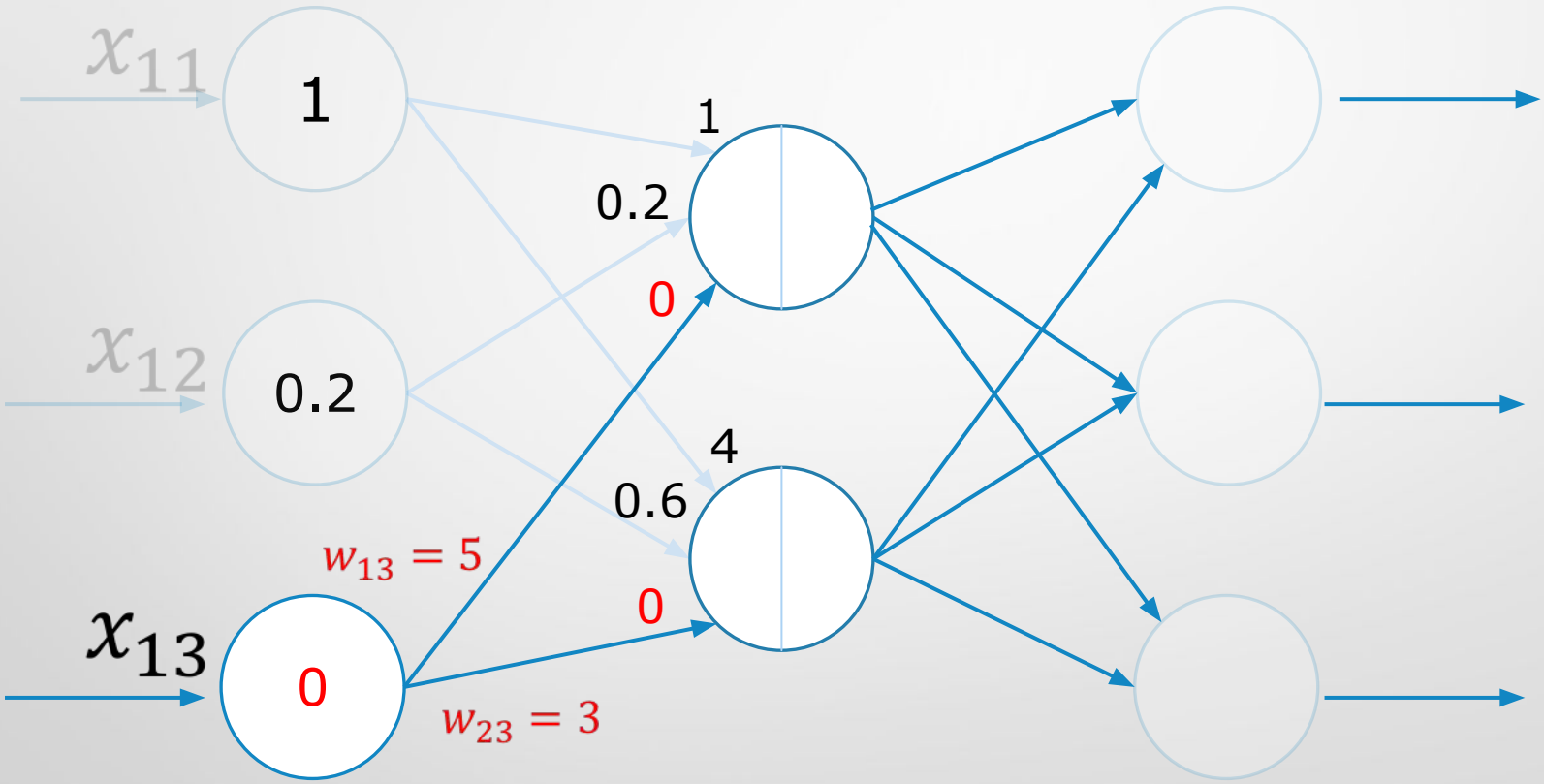
計算の流れ



重み行列

$$W = \begin{pmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 5 \\ 4 & 3 & 3 \end{pmatrix}$$

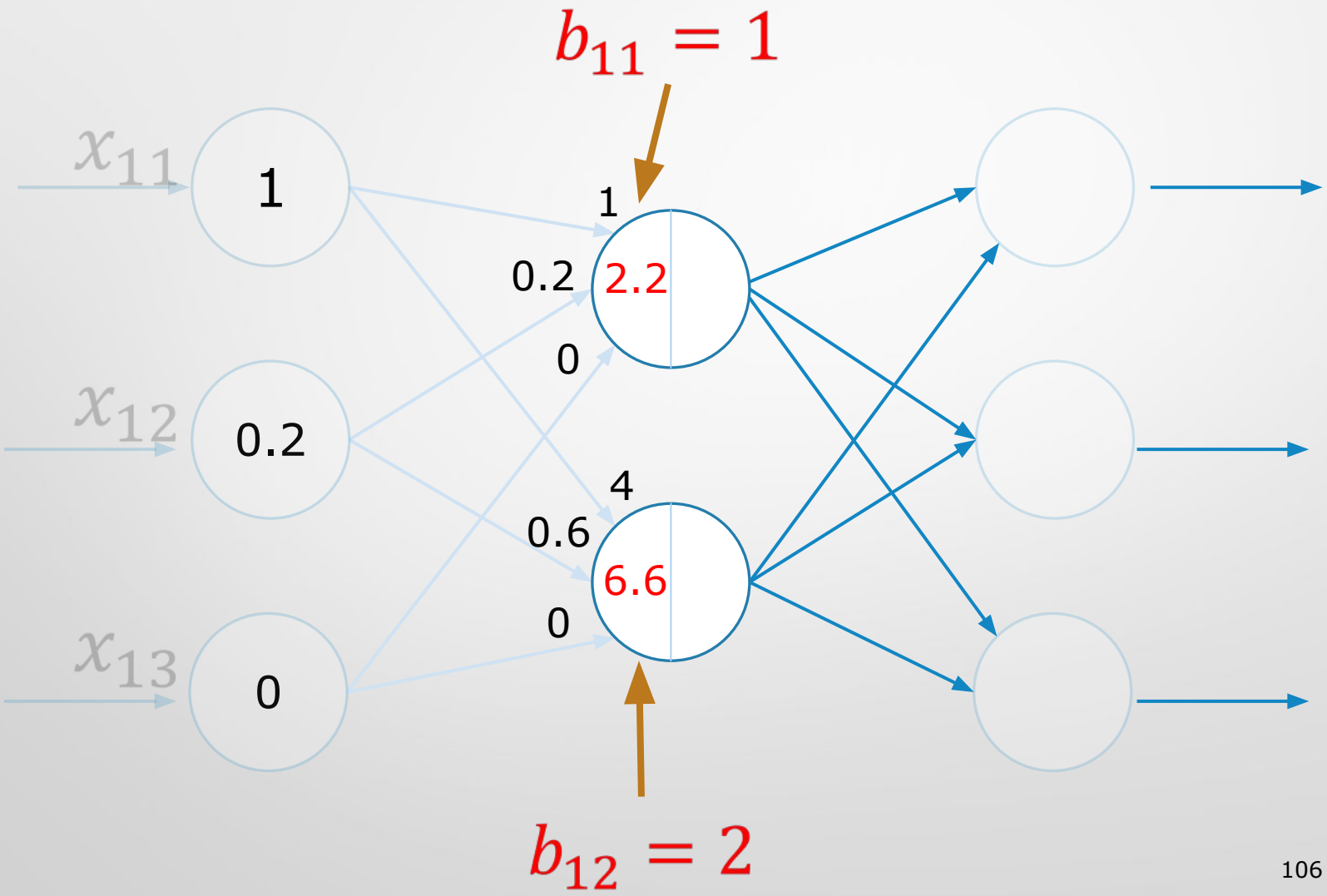
計算の流れ



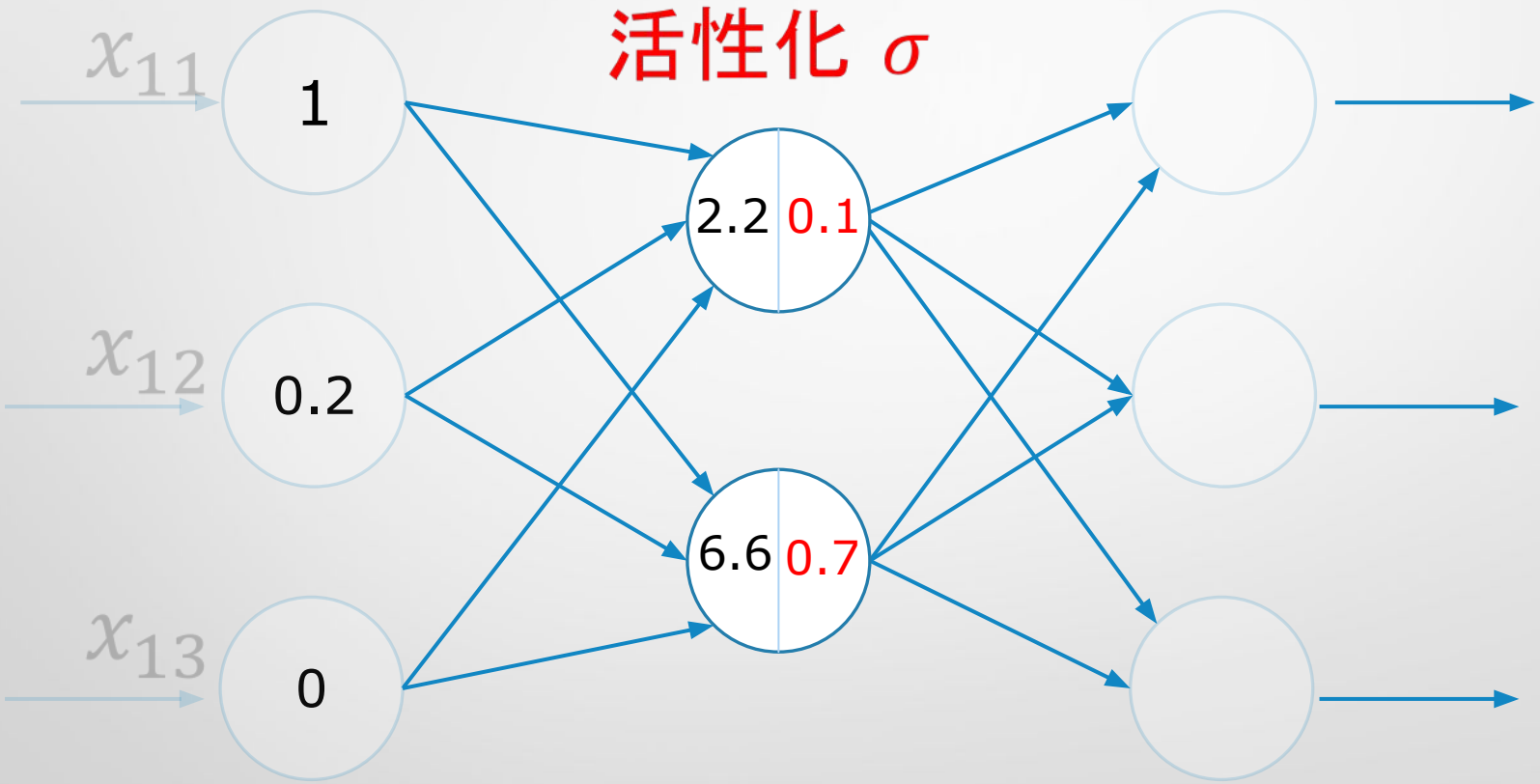
重み行列

$$W = \begin{pmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 5 \\ 4 & 3 & 3 \end{pmatrix}$$

計算の流れ

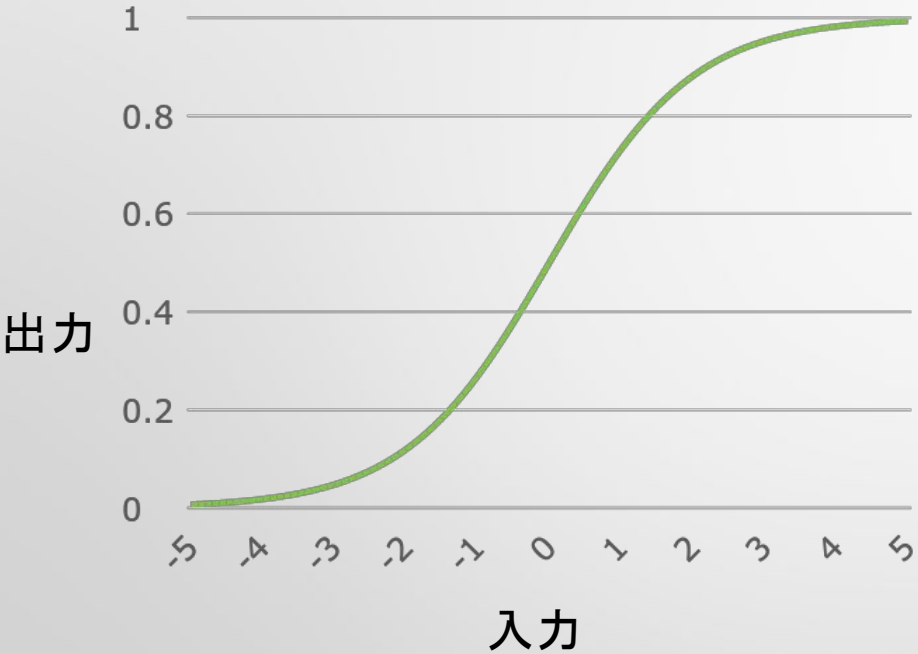


計算の流れ

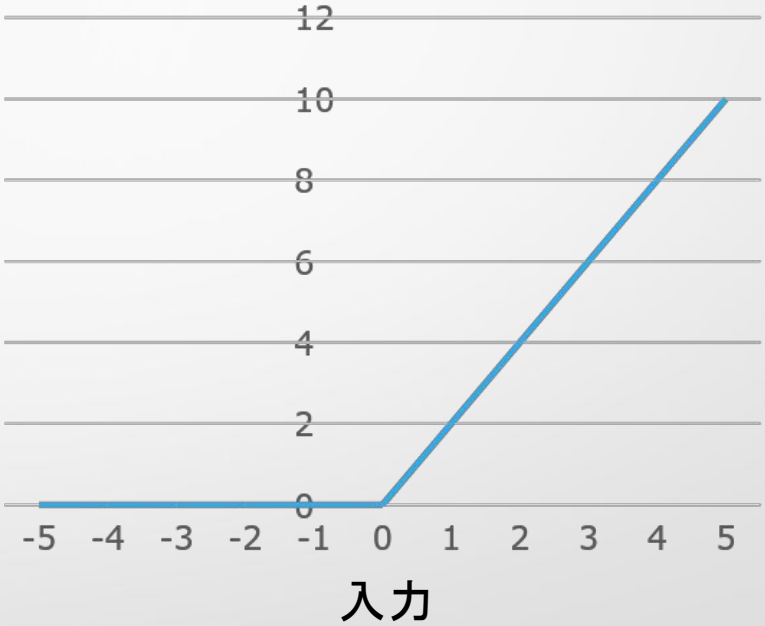


活性化関数

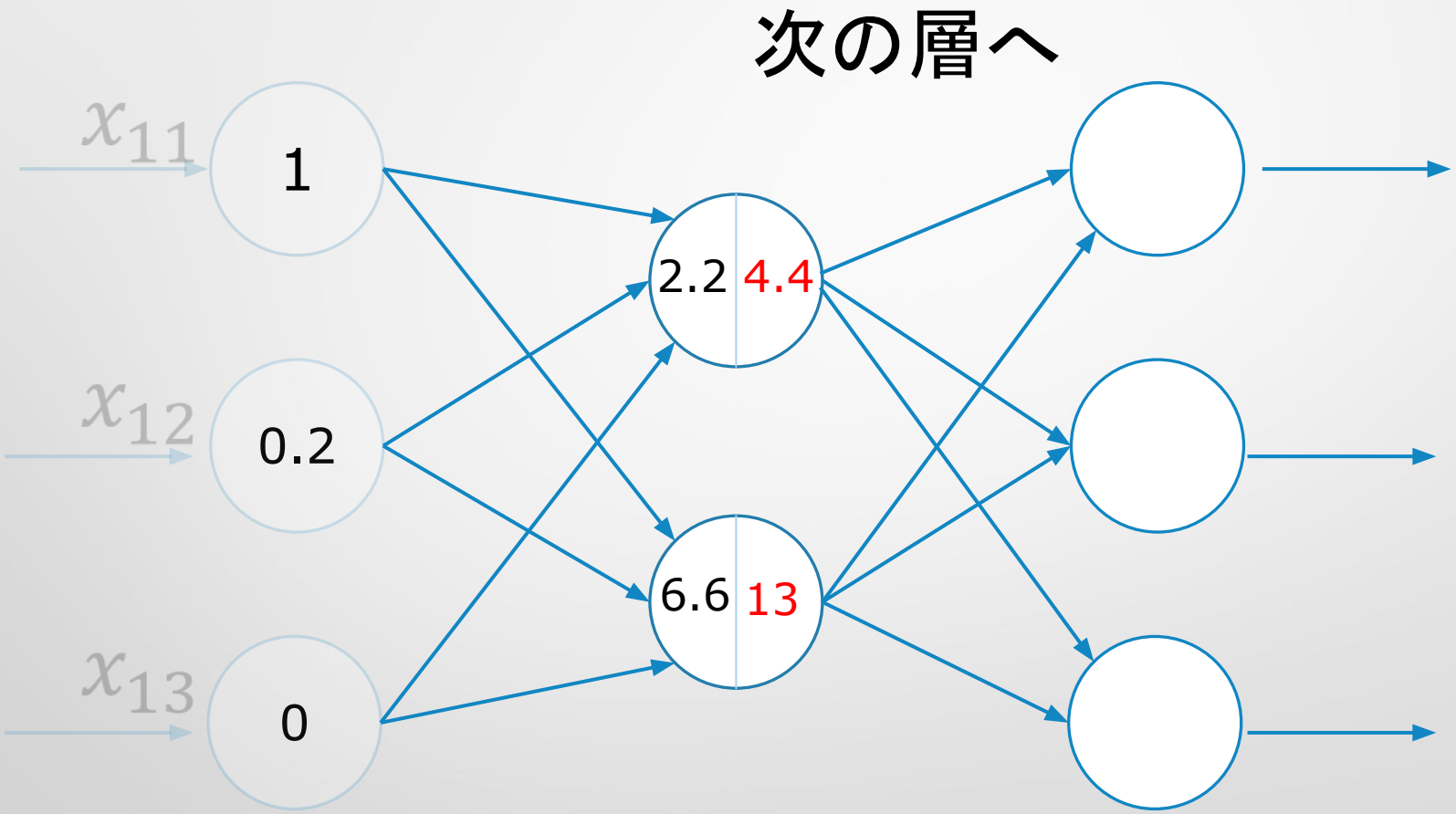
シグモイド関数



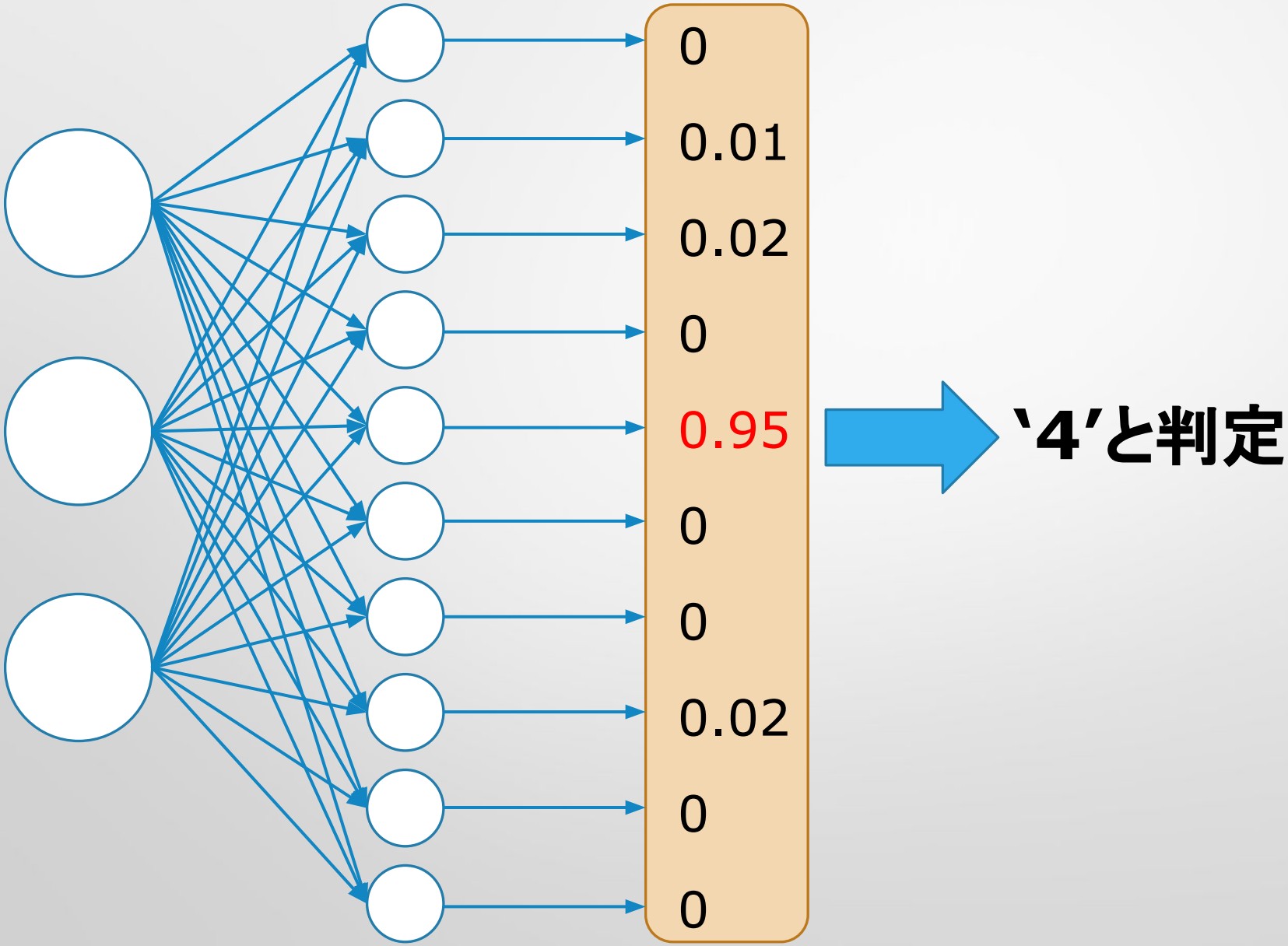
ReLU関数



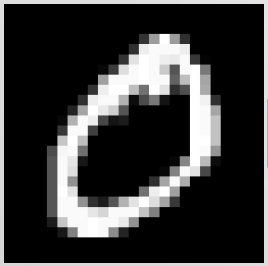
計算の流れ

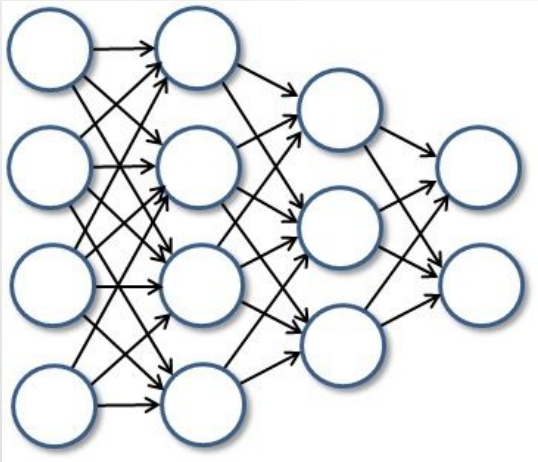


出力の形式



学習の流れ





0
0.01
0.02
0
0.95
0
0
0.02
0
0

‘4’
誤判定

学習の流れ

0	1
0.01	0
0.02	0
0	0
0.95	0
0	0
0	0
0.02	0
0	0
0	0

誤差計算
(損失関数)

重み行列

$$\begin{pmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 1 & 5 \\ 4 & 3 & 3 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 3 & 1 \\ 2 & 5 & 3 \end{pmatrix}$$

バイアス

$$\begin{pmatrix} b_{11} \\ b_{12} \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix} \rightarrow \begin{pmatrix} 1.1 \\ 3 \end{pmatrix}$$

誤差を減らすように
重み・バイアスを更新

学習の流れ

0	1
0.01	0
0.02	0
0	0
0.95	0
[illegible]	0
[illegible]	0
0.02	0
0	0
0	0

誤差計算
(損失関数)

二乗誤差
クロスエントロピー

誤差逆伝播

重み行列

$$\begin{pmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 5 \\ 4 & 3 & 3 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 3 & 1 \\ 2 & 5 & 3 \end{pmatrix}$$

バイアス

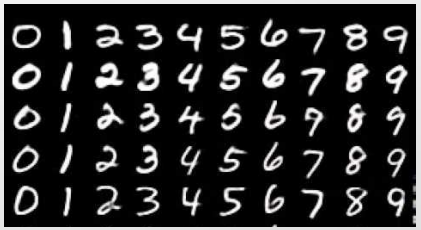
$$\begin{pmatrix} b_{11} \\ b_{12} \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix} \rightarrow \begin{pmatrix} 1.1 \\ 3 \end{pmatrix}$$

誤差を減らすように
重み・バイアスを更新

訓練データ・テストデータ

- データセットは2種類に分けて使う
- 訓練データ
  - 学習に用いるデータ
  - 更にバッチ(小分けにしたデータ)に分割
- テストデータ
  - 学習後の性能評価に用いる

データセット
60,000枚

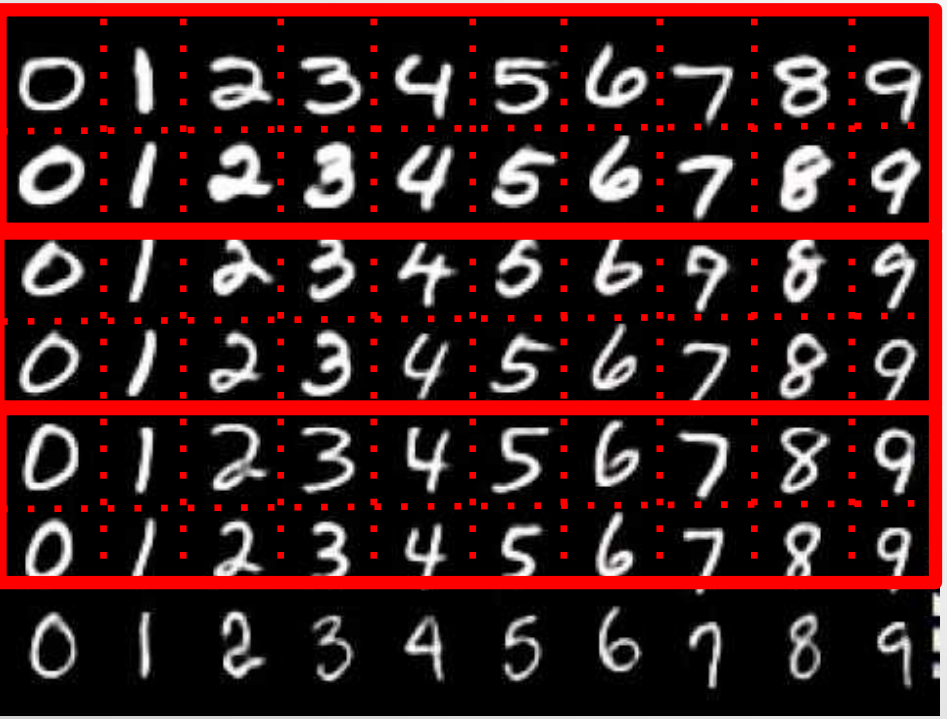


訓練データ
50,000枚

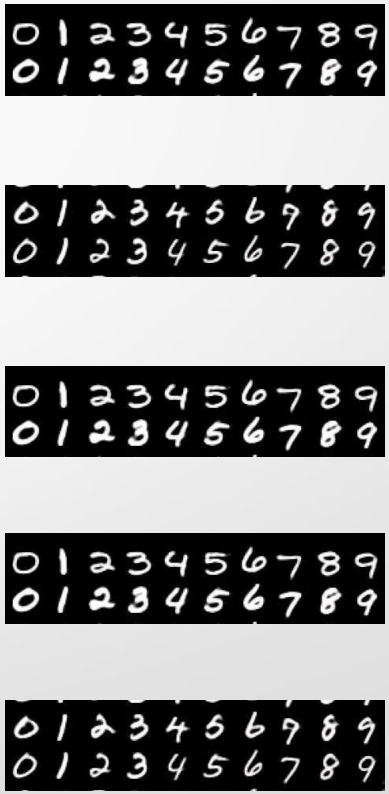
テストデータ
10,000枚

バッチサイズ

訓練データ
（例：50,000枚）

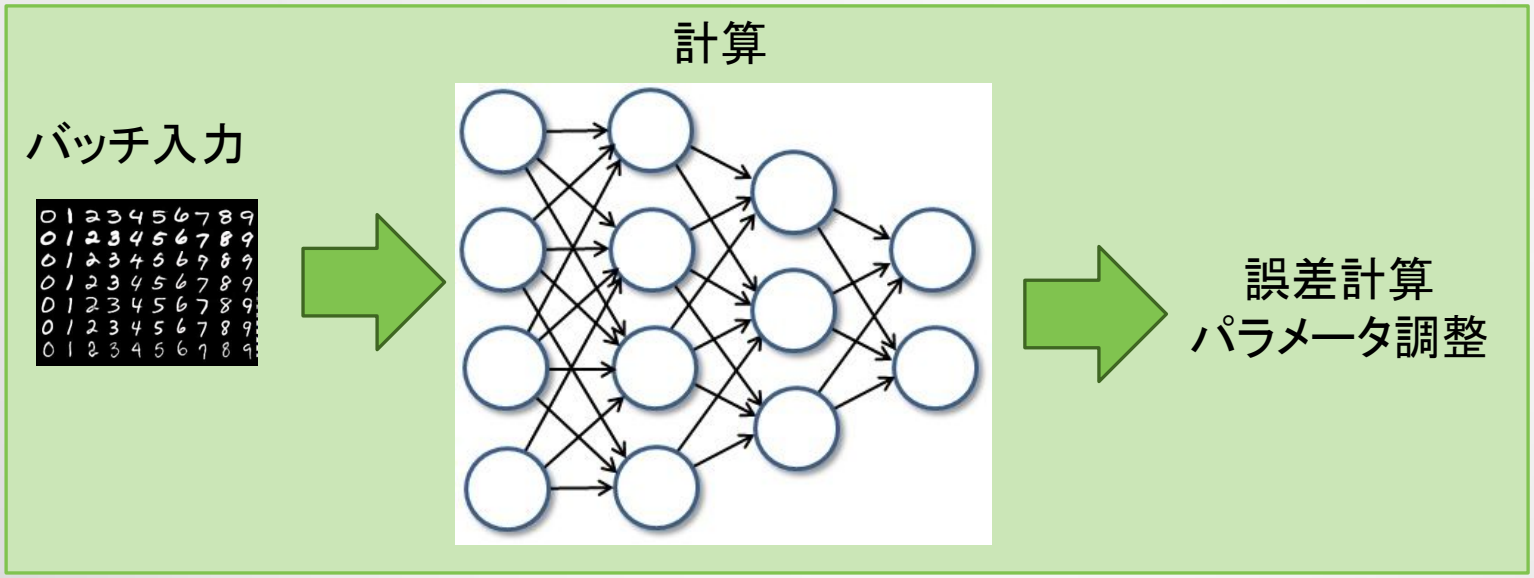


バッチ
（例：各10,000枚）



1度に入力するデータサイズ ＝ バッチサイズ

エポック



×

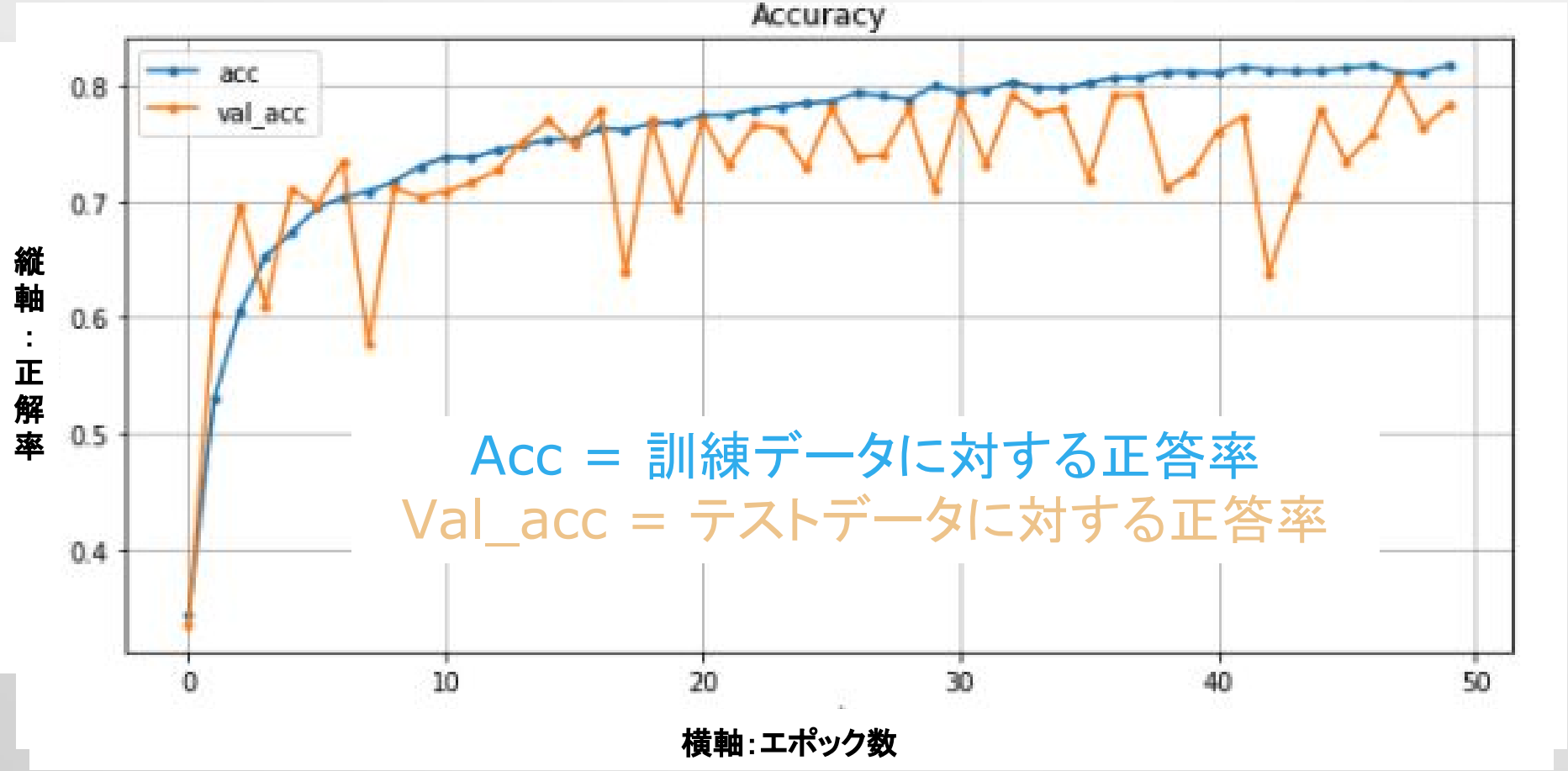
バッチ数回

||

1 エポック

訓練データ全体を読み込んだら1エポック

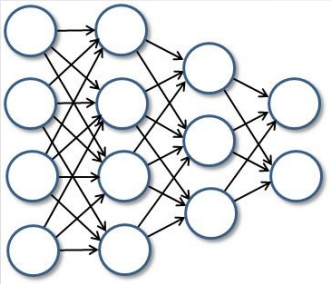
学習結果の評価



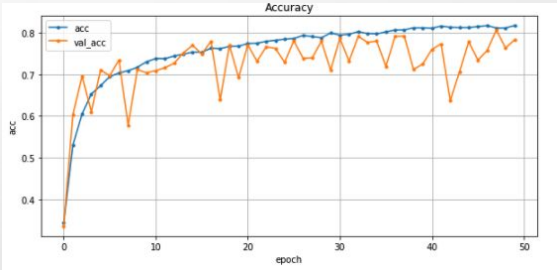
val_accは，テストデータで評価
->高いほど新しいデータに強い(性能のいいAI)

まとめ

データ選定・整形



パラメータ調整



精度評価

まとめ

- データ処理
  - バッチサイズ
  - エポック数
- ネットワーク構成
  - ユニット数(ニューロンの数)
  - レイヤー数(層の数)

→ **人間が設定するパラメータ**

- 計算パラメータ
  - 重み
  - バイアス

コード解説

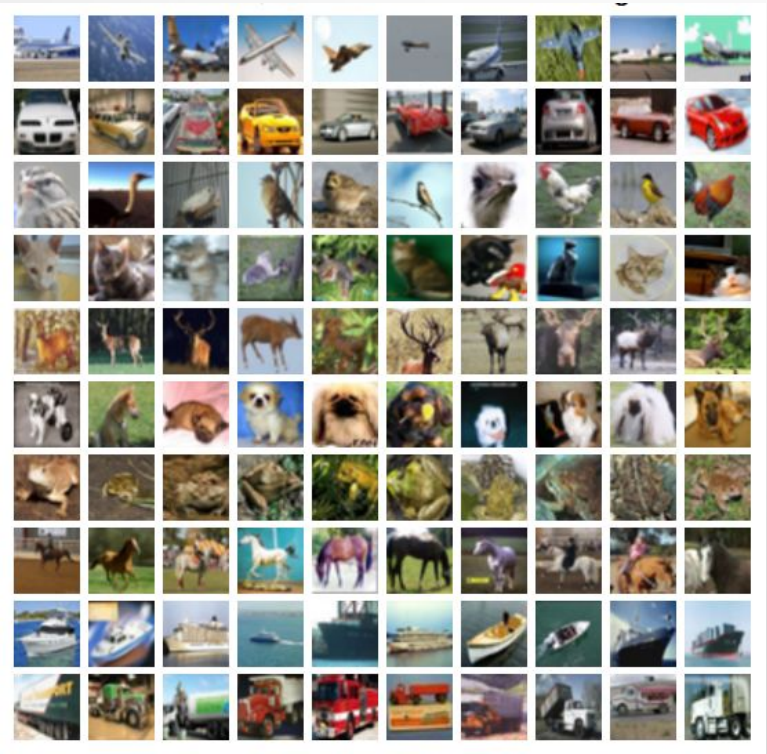
- パラメータチューニングのデモを行います

テスト

- 自分の手書き文字を入力してみましょう

フリーワーク

- CIFAR 100を使って，自由に画像分類
- CIFAR100
  - http://www.cs.toronto.edu/~kriz/cifar.html



発表タイム

- 使ったデータセット
- レイヤーアレンジ
- 精度，デモ

実際の活用指針

- Amazon AI
  - Amazon Rekognition
  - Amazon Polly
  - Amazon Lex
- Azure Cognitive Services API
  - Computer Vision
  - Speaker Recognition
  - Recommendations

まとめ

- EC2&Jupyter NotebookによるAIサーバ構築
- Kerasを用いたディープラーニング実装
- 画像の分類と評価
  - ニューロンとレイヤー
  - バッチサイズとエポック数
  - 精度の評価
- 体系的な理論
  - 活性化関数，損失関数，逆伝播法
  - 学習の仕組み

アンケート

- ご協力お願いします

Aidemyとは？

- ８月～リリース予定のAidemyをご紹介します！