

Etsuji Nakai
Cloud Solutions Architect at Google
2016/08/19 ver1.1

\$ who am i

- Etsuji Nakai

Cloud Solutions Architect at Google

Twitter @enakai00



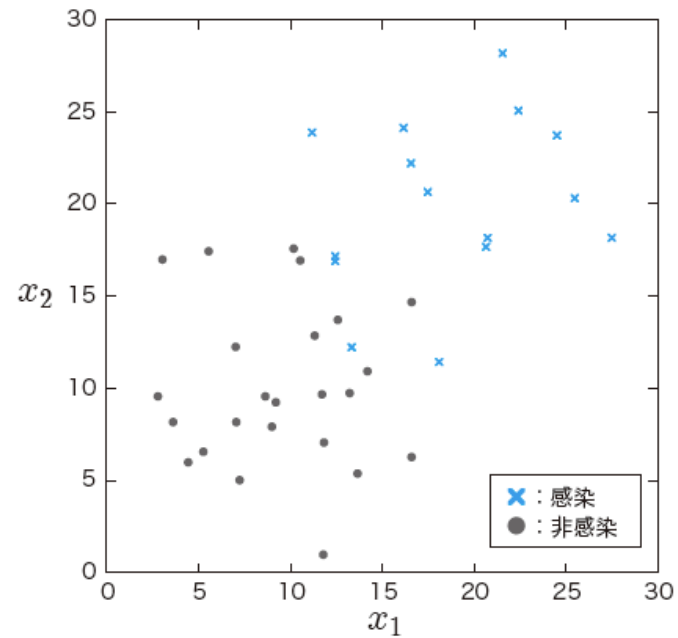


機械学習の基礎

■

●

<http://goo.gl/A2G4Hv>



■

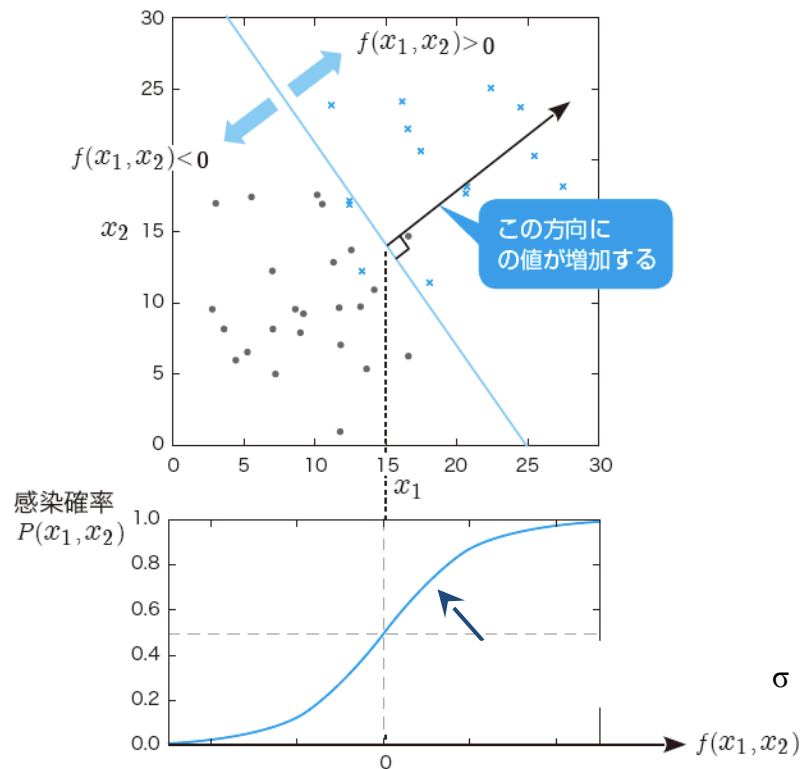
σ

$$f(x_1, x_2) = w_0 + w_1x_1 + w_2x_2$$

$$P(x_1, x_2) = \sigma(f(x_1, x_2))$$

■

$$(w_0, w_1, w_2)$$



σ

■

•

• $n \qquad (x_{1n}, x_{2n}) \qquad P_n$

$$t_n = 0, 1$$

$$P_n = \{P(x_{1n}, x_{2n})\}^{t_n} \{1 - P(x_{1n}, x_{2n})\}^{1-t_n}$$

•

$$P = P_1 \times P_2 \times \cdots \times P_N$$

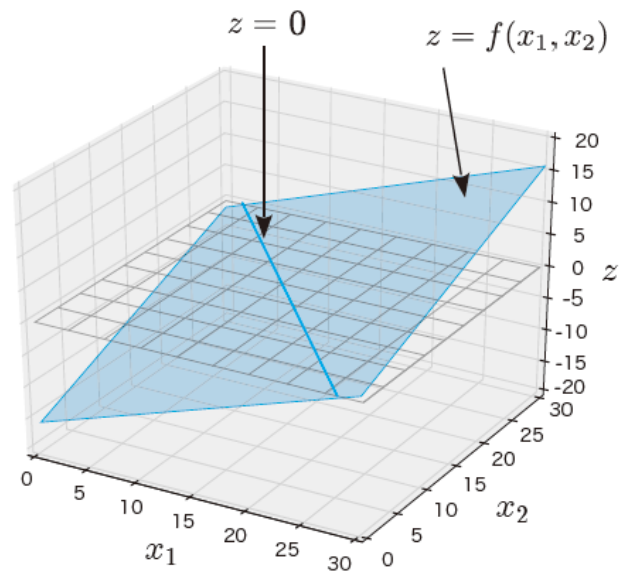
•

$$E = -\log P = \cdots = -\sum_{n=1}^N [t_n \log P(x_{1n}, x_{2n}) + (1 - t_n) \log \{1 - P(x_{1n}, x_{2n})\}]$$

■

$$z = f(x_1, x_2)$$

$$(x_1, x_2)$$



■

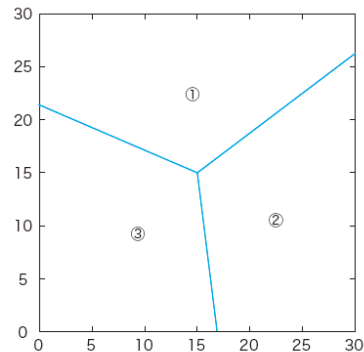
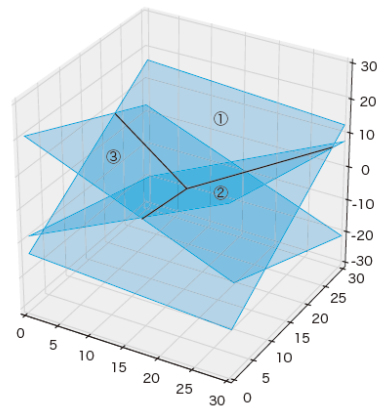
■

$$f_1(x_1, x_2) = w_{01} + w_{11}x_1 + w_{21}x_2$$

$$f_2(x_1, x_2) = w_{02} + w_{12}x_1 + w_{22}x_2$$

$$f_3(x_1, x_2) = w_{03} + w_{13}x_1 + w_{23}x_2$$

●



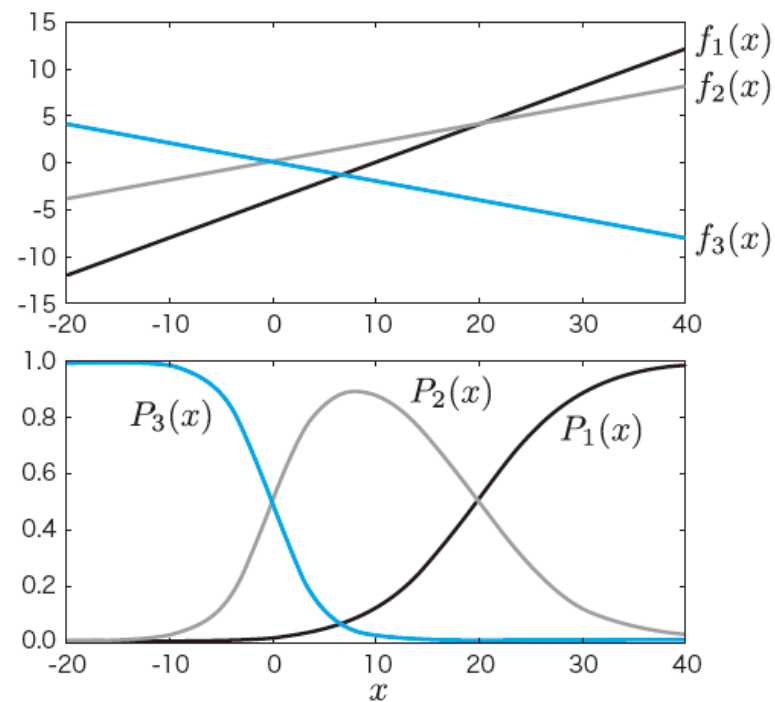
- $(x_1, x_2) \quad i$


$$P_i(x_1, x_2) = \frac{e^{f_i(x_1, x_2)}}{e^{f_1(x_1, x_2)} + e^{f_2(x_1, x_2)} + e^{f_3(x_1, x_2)}}$$

- f_1, f_2, f_3

$$0 \leq P_i \leq 1 \quad (i = 1, 2, 3)$$

$$P_1 + P_2 + P_3 = 1$$



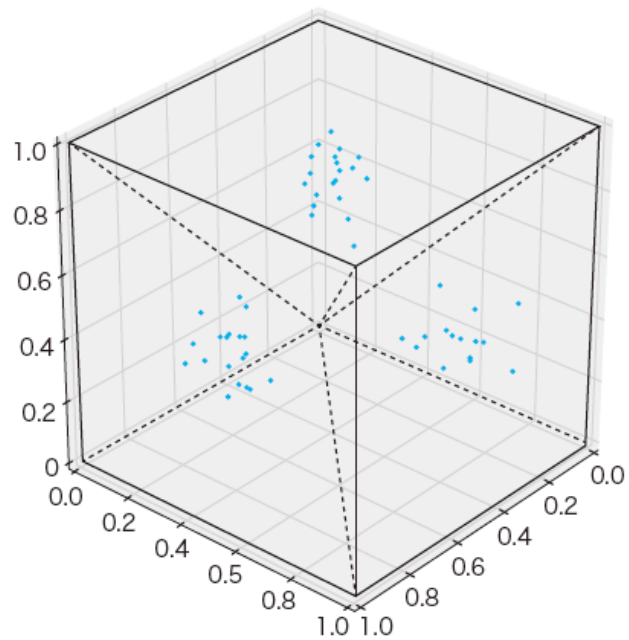


ニューラルネットワークによる 画像分類

■

■

•



[MSE-01] モジュールをインポートして、乱数のシードを設定します。

```
In [1]: import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.examples.tutorials.mnist import input_data

np.random.seed(20160604)
```

[MSE-02] MNISTのデータセットを用意します。

```
In [2]: mnist = input_data.read_data_sets("/tmp/data/", one_hot=True)

Extracting /tmp/data/train-images-idx3-ubyte.gz
Extracting /tmp/data/train-labels-idx1-ubyte.gz
Extracting /tmp/data/t10k-images-idx3-ubyte.gz
Extracting /tmp/data/t10k-labels-idx1-ubyte.gz
```

[MSE-03] ソフトマックス関数による確率 p の計算式を用意します。

```
In [3]: x = tf.placeholder(tf.float32, [None, 784])
w = tf.Variable(tf.zeros([784, 10]))
w0 = tf.Variable(tf.zeros([10]))
f = tf.matmul(x, w) + w0
p = tf.nn.softmax(f)
```

[MSE-04] 誤差関数 loss とトレーニングアルゴリズム train_step を用意します。

```
In [4]: t = tf.placeholder(tf.float32, [None, 10])
loss = -tf.reduce_sum(t * tf.log(p))
train_step = tf.train.AdamOptimizer().minimize(loss)
```

[MSE-05] 正解率 accuracy を定義します。

```
In [5]: correct_prediction = tf.equal(tf.argmax(p, 1), tf.argmax(t, 1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
```

[MSE-06] セッションを用意して、Variableを初期化します。

```
In [6]: sess = tf.InteractiveSession()
sess.run(tf.initialize_all_variables())
```

[MSE-07] パラメーターの最適化を2000回繰り返します。

1回の処理において、トレーニングセットから取り出した100個のデータを用いて、勾配降下法を適用します。

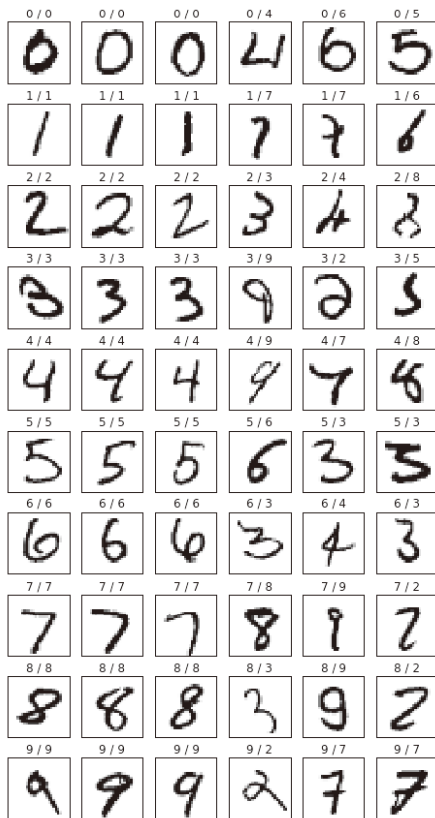
最終的に、テストセットに対して約92%の正解率が得られます。

```
In [7]: i = 0
for _ in range(2000):
    i += 1
    batch_xs, batch_ts = mnist.train.next_batch(100)
    sess.run(train_step, feed_dict={x: batch_xs, t: batch_ts})
    if i % 100 == 0:
        loss_val, acc_val = sess.run([loss, accuracy],
                                     feed_dict={x: mnist.test.images, t: mnist.test.labels})
        print('Step: %d, Loss: %f, Accuracy: %f'
              % (i, loss_val, acc_val))
```

```
Step: 100, Loss: 7747.077148, Accuracy: 0.848400
Step: 200, Loss: 5439.362305, Accuracy: 0.879900
Step: 300, Loss: 4556.467285, Accuracy: 0.890900
Step: 400, Loss: 4132.035156, Accuracy: 0.896100
Step: 500, Loss: 3836.139160, Accuracy: 0.902600
Step: 600, Loss: 3646.572510, Accuracy: 0.903900
Step: 700, Loss: 3490.270752, Accuracy: 0.909100
Step: 800, Loss: 3385.605469, Accuracy: 0.909400
Step: 900, Loss: 3293.132324, Accuracy: 0.912800
Step: 1000, Loss: 3220.884277, Accuracy: 0.913700
Step: 1100, Loss: 3174.230957, Accuracy: 0.913700
Step: 1200, Loss: 3081.114990, Accuracy: 0.916400
Step: 1300, Loss: 3046.678711, Accuracy: 0.915400
Step: 1400, Loss: 3002.018555, Accuracy: 0.916300
Step: 1500, Loss: 2973.873779, Accuracy: 0.918700
Step: 1600, Loss: 2960.562500, Accuracy: 0.918200
Step: 1700, Loss: 2923.289062, Accuracy: 0.917500
Step: 1800, Loss: 2902.116699, Accuracy: 0.919000
Step: 1900, Loss: 2870.737061, Accuracy: 0.920000
Step: 2000, Loss: 2857.827881, Accuracy: 0.921100
```

正解例

不正解例



<http://goo.gl/rGqjYh>

TensorFlowで学ぶディープラーニング入門
~畳み込みニューラルネットワーク徹底解説~

単行本（ソフトカバー） - 2016/9/28

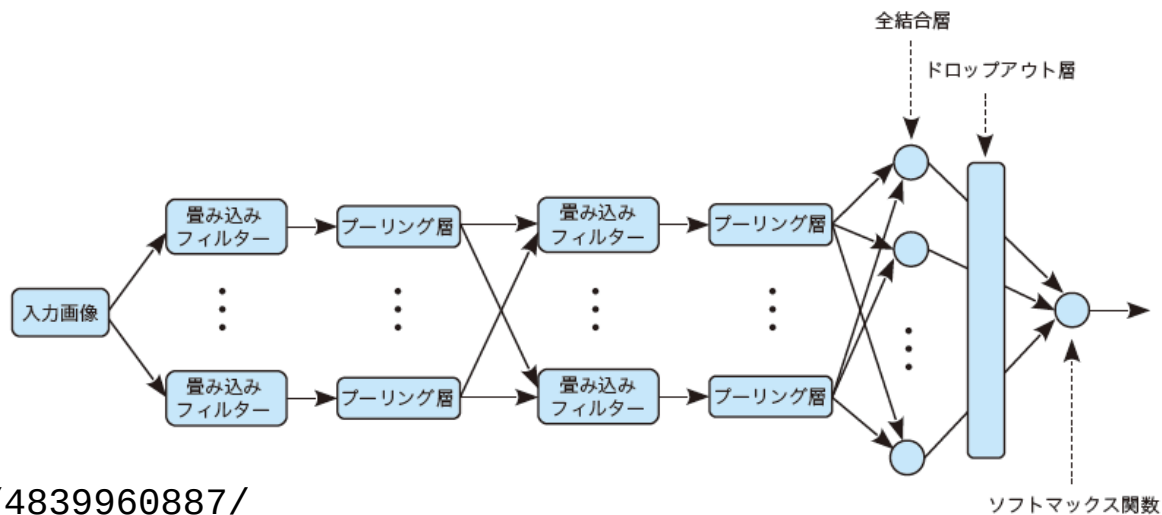
中井 悦司 (著)

▶ その他 () の形式およびエディションを表示する

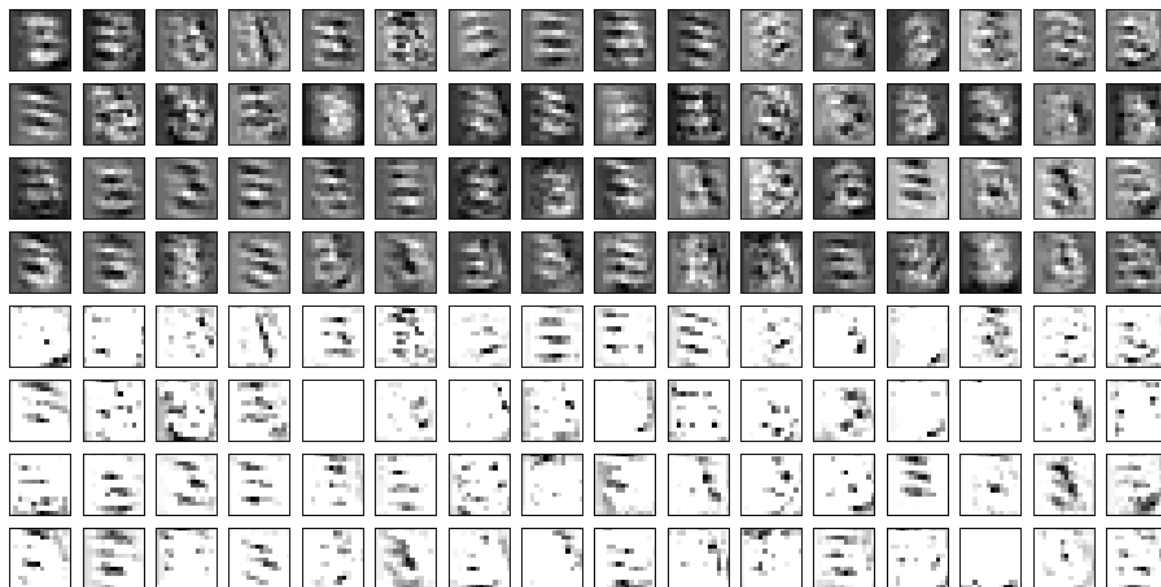
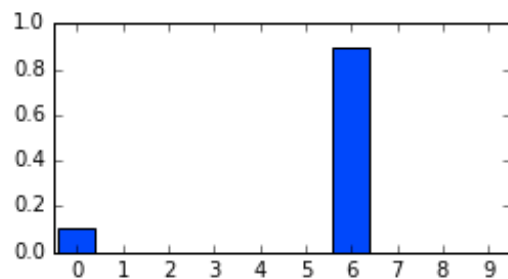
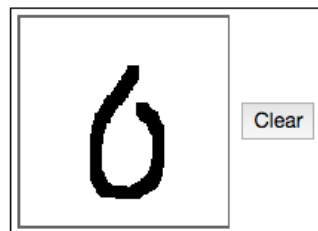
単行本（ソフトカバー）

¥ 2,905  プライム

¥ 2,905 より 1 新品

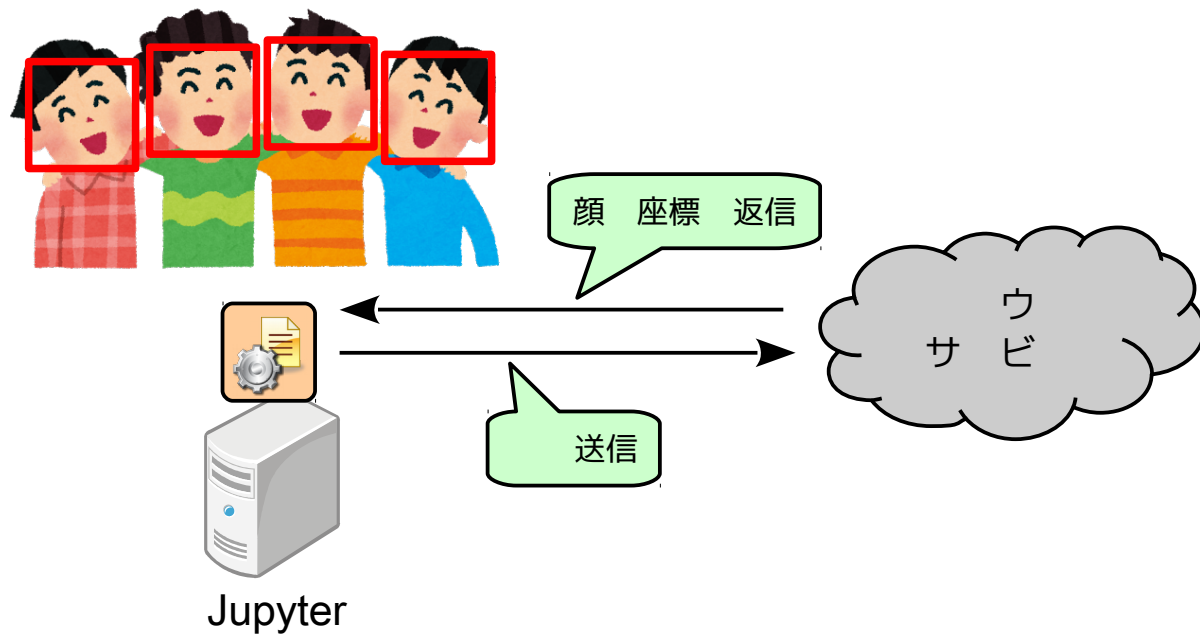


<https://www.amazon.co.jp/dp/4839960887/>



■ <http://goo.gl/UHsVmI>
<http://goo.gl/VE2ISf>

A I サ ビ



■ <http://goo.gl/dh6cwB>



機械学習を利用した クライアントアプリケーションの例

A I サ ビ

利

- ブ ウザ コ カ 取得
ウ A I サ ビ 送信 笑顔 識別



<http://goo.gl/9EM8tr>

サバ別施

■ Li サ バ ズ 利 ゆ 自動仕

キュウリ農家とディープラーニングをつなぐ TensorFlow

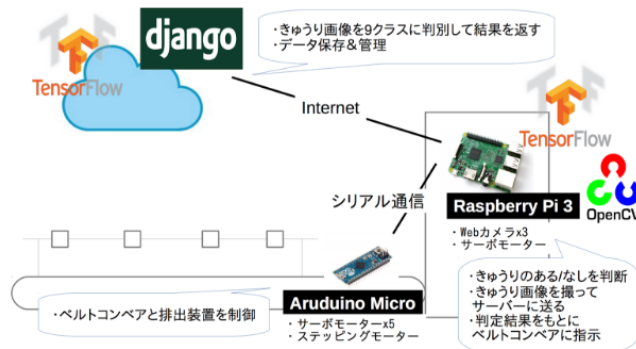
2016年8月5日金曜日

Posted by 佐藤一憲 (Cloud Platform チーム デベロッパーアドボケイト)

「Google のコンピュータ囲碁プログラム『AlphaGo』が世界トップクラスの棋士と互角に指し合う様子を見て、これは凄いことが起きている、と思いました。それが、ディープラーニングを使ったキュウリ仕分け機の開発のきっかけです」

それからわずか4か月。静岡県でキュウリ栽培農家を営む小池誠さんは、Google の機械学習ライブラリ TensorFlow を用いた**キュウリ仕分け機**の自作を進め、その試作2号機が7月に完成した。8月6日～7日に東京ビッグサイトで開催されるイベント

「Maker Faire Tokyo 2016」で展示される。

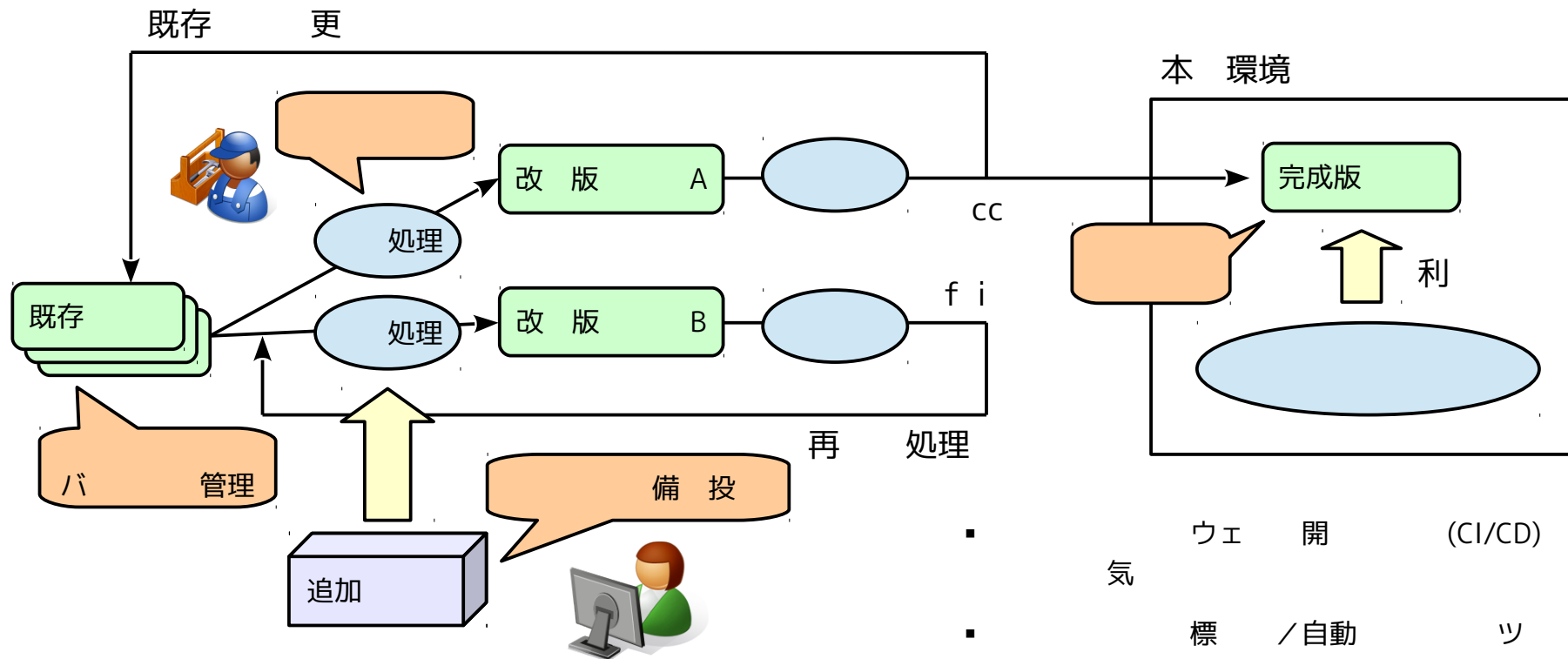


http://googlecloudplatform-japan.blogspot.jp/2016/08/tensorflow_5.html

他 可

- べ 個別 手 子
- ・ カ コ 言 転移 手法
- ウ 接 行
- ・ 子 行 だ 側 対応可 必要
- ム 処理 施
- ・ 簡単 取得 処理 十 何 白

適





Google Cloud Platform

Thank you!