機械学習勉強会2.0

時系列解析

Academeia

講師紹介

野田　真史　博士（工学）

2007年に博士課程を修了。

分子科学研究所、筑波大学で博士研究員を経た後、2020年にアカデメイア入社。

これまでの専門は第一原理計算を用いた物性予測。

SALMON(電子ダイナミクス計算プログラム)の講習会の講師を2度務める。

G検定所持。

RNN、LSTMの概要

RNN (Recurrent Neural Network)

$$h_j = \tanh(W x_j + U h_{j-1} + b)$$

h_j : 中間層からの出力

W : 入力層から中間層への重み

x_j : 入力

U : 中間層から中間層への重み

b : バイアス

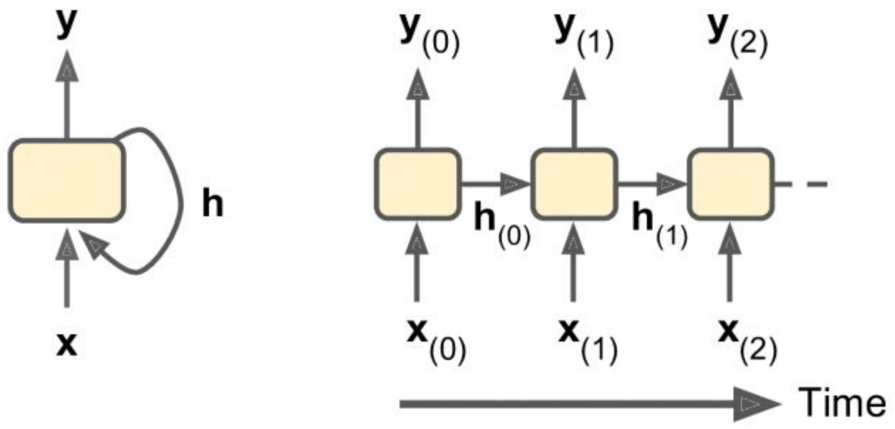


Fig. 15-3 (Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, Aurélien Géron)

勾配消失問題

$$\frac{\partial E_t}{\partial W} = \sum_{k=0}^{T} \frac{\partial E_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} \left(\prod_{j=t+1}^{T} \frac{\partial h_j}{\partial h_{j-1}} \right) \frac{\partial h_t}{\partial W}$$

E_t ：誤差

$$h_j = \tanh(Wx_j + Uh_{j-1} + b)$$

$$\frac{\partial h_j}{\partial h_{j-1}} = U\left(1 - \tanh^2(Wx_j + Uh_{j-1} + b)\right)$$

$\left| \frac{\partial h_j}{\partial h_{j-1}} \right| < 1$ のためTが大きくなると誤差が消失する。

LSTM (Long Short Term Memory)

$$
\begin{aligned}
\mathbf{i}_{(t)} &= \sigma(\mathbf{W}_{xi}{}^{\top}\mathbf{x}_{(t)} + \mathbf{W}_{hi}{}^{\top}\mathbf{h}_{(t-1)} + \mathbf{b}_i) \\
\mathbf{f}_{(t)} &= \sigma(\mathbf{W}_{xf}{}^{\top}\mathbf{x}_{(t)} + \mathbf{W}_{hf}{}^{\top}\mathbf{h}_{(t-1)} + \mathbf{b}_f) \\
\mathbf{o}_{(t)} &= \sigma(\mathbf{W}_{xo}{}^{\top}\mathbf{x}_{(t)} + \mathbf{W}_{ho}{}^{\top}\mathbf{h}_{(t-1)} + \mathbf{b}_o) \\
\mathbf{g}_{(t)} &= \tanh(\mathbf{W}_{xg}{}^{\top}\mathbf{x}_{(t)} + \mathbf{W}_{hg}{}^{\top}\mathbf{h}_{(t-1)} + \mathbf{b}_g) \\
\mathbf{c}_{(t)} &= \mathbf{f}_{(t)} \otimes \mathbf{c}_{(t-1)} + \mathbf{i}_{(t)} \otimes \mathbf{g}_{(t)} \\
\mathbf{y}_{(t)} &= \mathbf{h}_{(t)} = \mathbf{o}_{(t)} \otimes \tanh(\mathbf{c}_{(t)})
\end{aligned}
$$

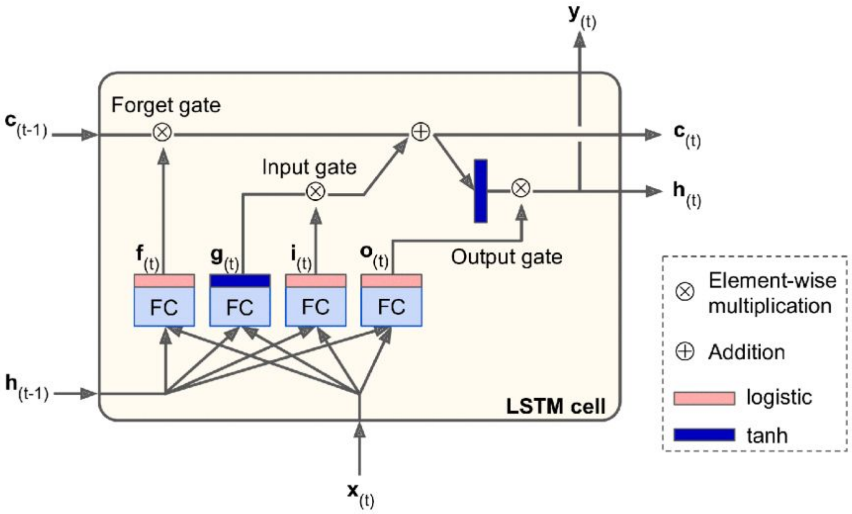


Fig. 15-9 (Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, Aurélien Géron)

忘却ゲートを設けることにより勾配消失問題を解消。

LSTM演習

Lesson 1: dataset - ダウンロード

```
zip_path = tf.keras.utils.get_file(
    origin='https://storage.googleapis.com/tensorflow/
                tf-keras-datasets/jena_climate_2009_2016.csv.zip',
    fname='jena_climate_2009_2016.csv.zip',
    extract=True)
csv_path, _ = os.path.splitext(zip_path)
```

Max Planck Institute for Biogeochemistry の weather time series dataset。
2003年から観測されている。
François Chollet が彼の著書 Deep Learning with Python のために2009年から2016年までのデータをPython用に編集。

Lesson 1: dataset - データの表示

```
df = pd.read_csv(csv_path)
df.head()
```

Out[1]:

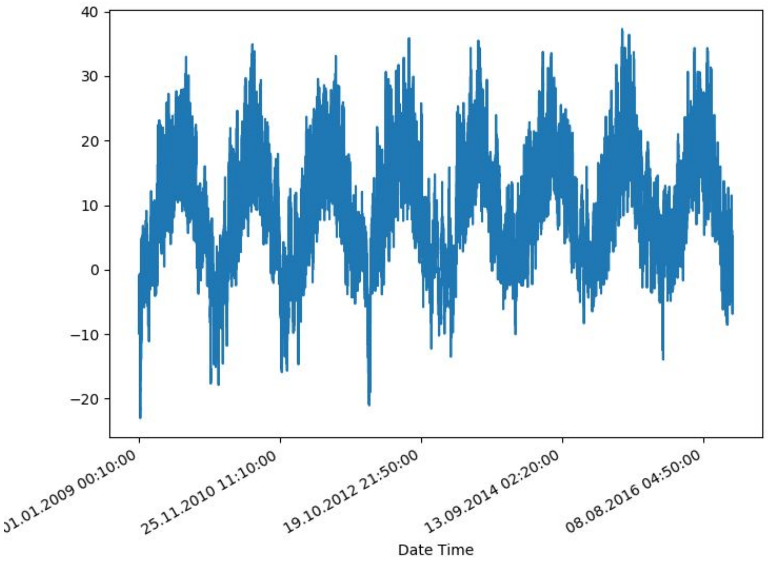
	Date Time	p (mbar)	T (degC)	Tpot (K)	Tdew (degC)	rh (%)	VPmax (mbar)	VPact (mbar)	VPdef (mbar)	sh (g/kg)	H2OC (mmol/mol)	rho (g/m**3)	wv (m/s)	max. wv (m/s)	wd (deg)
0	01.01.2009 00:10:00	996.52	-8.02	265.40	-8.90	93.3	3.33	3.11	0.22	1.94	3.12	1307.75	1.03	1.75	152.3
1	01.01.2009 00:20:00	996.57	-8.41	265.01	-9.28	93.4	3.23	3.02	0.21	1.89	3.03	1309.80	0.72	1.50	136.1
2	01.01.2009 00:30:00	996.53	-8.51	264.91	-9.31	93.9	3.21	3.01	0.20	1.88	3.02	1310.24	0.19	0.63	171.6
3	01.01.2009 00:40:00	996.51	-8.31	265.12	-9.07	94.2	3.26	3.07	0.19	1.92	3.08	1309.19	0.34	0.50	198.0
4	01.01.2009 00:50:00	996.51	-8.27	265.15	-9.04	94.1	3.27	3.08	0.19	1.92	3.09	1309.00	0.32	0.63	214.3

14の特徴量が10分毎に観測されていることがわかる。

演習　コマンドライン：`$ python lesson1 dataset.py`
jupyter notebook：# lesson1_dataset の欄を Ctrl+Enter
(コマンドラインでは表示されない可能性があります。後の作業に影響はありません。)

Lesson 2: 単変量の時系列予測 - 気温のプロット

```
uni_data = df['T (degC)']
uni_data.plot(subplots=True)
```



演習　コマンドライン：`$ python lesson2 univariate baseline 1.py utils.py`
jupyter notebook：# lesson2_univariate_baseline_1 の欄を Ctrl+Enter

Lesson 2: 単変量の時系列予測 - データの変換

```
def univariate_data(dataset, start_index, end_index,
                    history_size,  target_size):
  data = []
  labels = []

  start_index = start_index + history_size
  if end_index is None:
    end_index = len(dataset) - target_size

  for i in range(start_index, end_index):
    indices = range(i-history_size, i)
    # Reshape data from (history_size,) to (history_size, 1)
    data.append(np.reshape(dataset[indices], (history_size, 1)))
    labels.append(dataset[i+target_size])
  return np.array(data), np.array(labels)
```

LSTM層に入力するため、datasetからnp.array(data), np.array(labels)を返す関数を定義。

Lesson 2: 単変量の時系列予測 - 前処理

```
uni_data = uni_data.values
uni_train_mean = uni_data[:TRAIN_SPLIT].mean()
uni_train_std = uni_data[:TRAIN_SPLIT].std()
uni_data = (uni_data-uni_train_mean)/uni_train_std
```

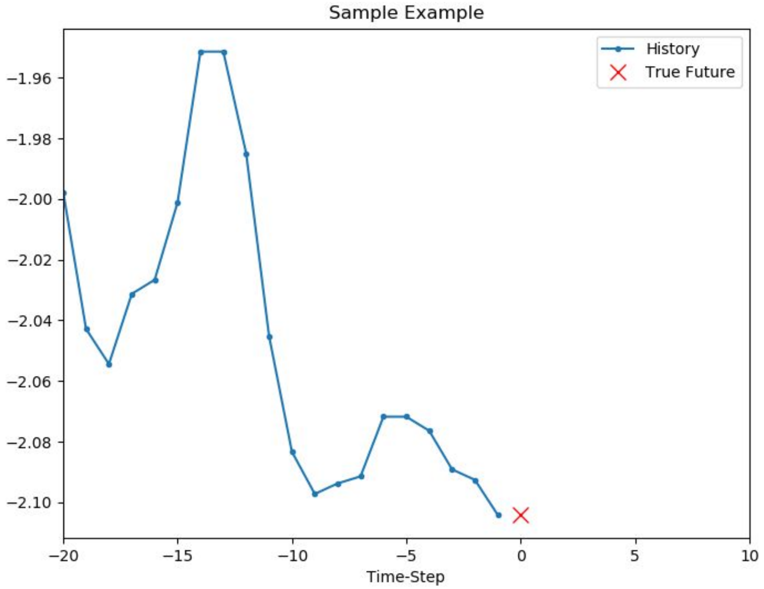
平均を引き、標準偏差で割る。

```
x_train_uni, y_train_uni = univariate_data(uni_data, 0, TRAIN_SPLIT,
                                           univariate_past_history,
                                           univariate_future_target)
x_val_uni, y_val_uni = univariate_data(uni_data, TRAIN_SPLIT, None,
                                       univariate_past_history,
                                       univariate_future_target)
```

LSTM層に渡せるよう、データの形式を変換。

Lesson 2: 単変量の時系列予測 - 前処理後のグラフ

```
show_plot([x_train_uni[0], y_train_uni[0]], 0, 'Sample Example')
```



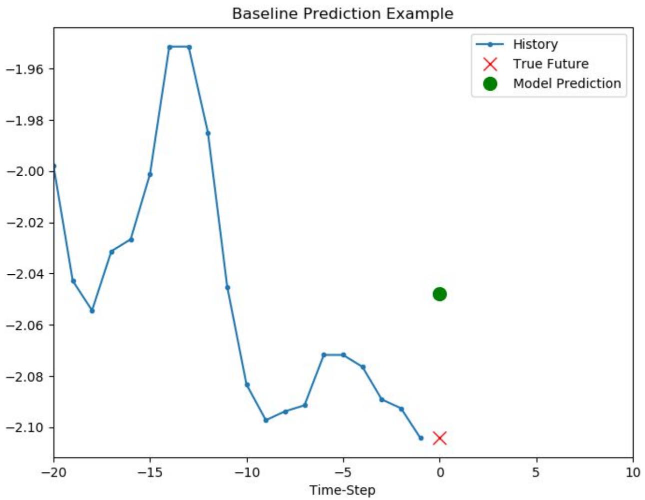
演習　コマンドライン：`$ python lesson2_univariate_baseline_2.py utils.py`
jupyter notebook：# lesson2_univariate_baseline_2 の欄を Ctrl+Enter

Lesson 2: 単変量の時系列予測 - baseline

```
def baseline(history):
  return np.mean(history)

show_plot([x_train_uni[0], y_train_uni[0],
           baseline(x_train_uni[0])], 0,
          'Baseline Prediction Example')
```

baselineを過去20点の平均で指定。



演習　コマンドライン：`$ python lesson2 univariate baseline 3.py utils.py`
jupyter notebook：# lesson2_univariate_baseline_3 の欄を Ctrl+Enter

Lesson 3: 単変量の時系列予測 - バッチ処理

```
BATCH_SIZE = 256
BUFFER_SIZE = 10000

train_univariate = tf.data.Dataset.from_tensor_slices((x_train_uni, y_train_uni))
train_univariate = \
  train_univariate.cache().shuffle(BUFFER_SIZE).batch(BATCH_SIZE).repeat()

val_univariate = tf.data.Dataset.from_tensor_slices((x_val_uni, y_val_uni))
val_univariate = val_univariate.batch(BATCH_SIZE).repeat()
```

バッチサイズ256、バッファサイズ10000でシャッフルされた訓練データと
バッチサイズ256の評価データを用意。

Lesson 3: 単変量の時系列予測 - 訓練

```
simple_lstm_model = tf.keras.models.Sequential([
  tf.keras.layers.LSTM(8, input_shape=x_train_uni.shape[-2:]),
  tf.keras.layers.Dense(1)
```

LSTM層(出力8次元)と、出力1次元の層(予測したい時刻の温度)のモデルを作成。

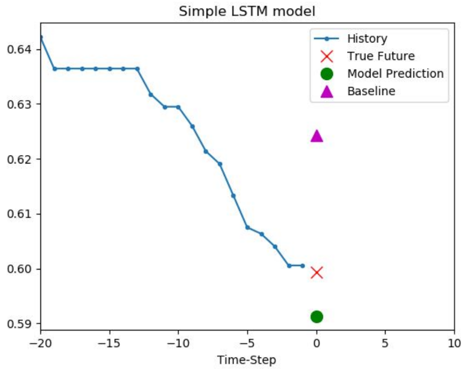
```
EVALUATION_INTERVAL = 200
EPOCHS = 10
simple_lstm_model.fit(train_univariate, epochs=EPOCHS,
                      steps_per_epoch=EVALUATION_INTERVAL,
                      validation_data=val_univariate, validation_steps=50)
```

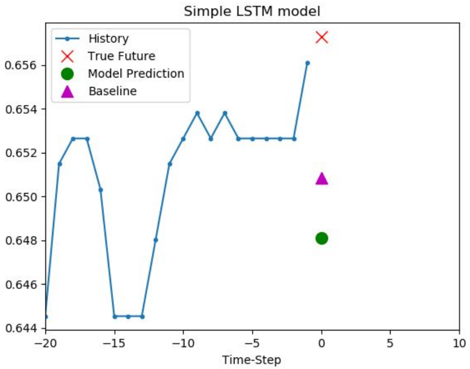
訓練データのステップ数 200、評価データのステップ数 50で訓練を行う。

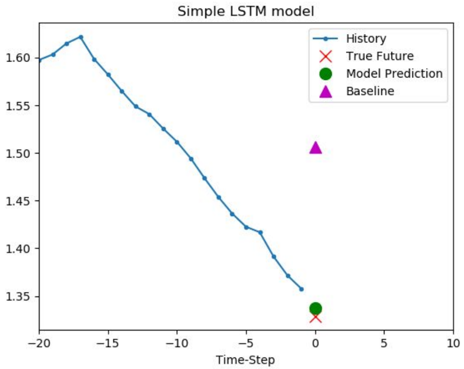
Lesson 3: 単変量の時系列予測 - 予測結果

```
for x, y in val_univariate.take(3):
  plot = show_plot_with_baseline([x[0].numpy(), y[0].numpy(),
  simple_lstm_model.predict(x)[0]], 0, 'Simple LSTM model')
  plot.show()
```

予測された気温の
3パターンが
表示される。





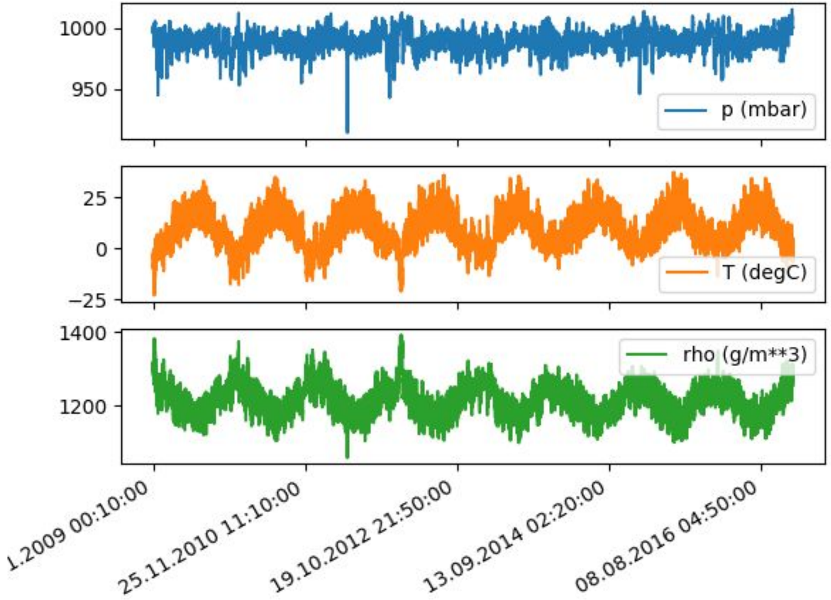


演習　コマンドライン：`$ python lesson3 univariate forecasting.py utils.py`
jupyter notebook：# lesson3_univariate_forecasting の欄を Ctrl+Enter

Lesson 4: 多変量の時系列予測 - 多変量

ここでは、気圧、気温、密度のデータからある時刻 (1点)の気温を予測する。

```
features_considered = ['p (mbar)', 'T (degC)', 'rho (g/m**3)']
features = df[features_considered]
features.plot(subplots=True)
```

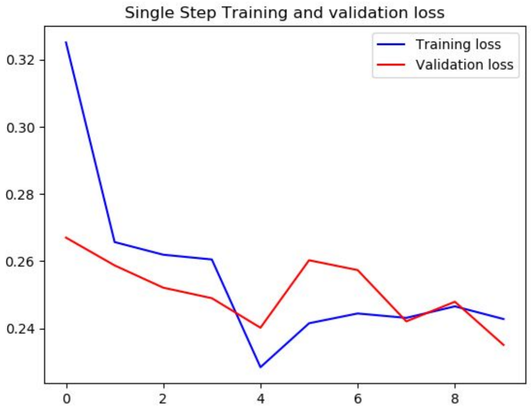


Lesson 4: 多変量の時系列予測 - 訓練

```
single_step_model = tf.keras.models.Sequential()
single_step_model.add(tf.keras.layers.LSTM(32,
                                           input_shape=x_train_single.shape[-2:]))
single_step_model.add(tf.keras.layers.Dense(1))
single_step_model.compile(optimizer=tf.keras.optimizers.RMSprop(),
                          loss='mae')
```

LSTM層の出力は32次元。optimizerはRMSprop。損失関数はMAE。

```
single_step_history = \
  single_step_model.fit(train_data_single,
                        epochs=EPOCHS,
                        steps_per_epoch=EVALUATION_INTERVAL,
                        validation_data=val_data_single,
                        validation_steps=50)
plot_train_history(single_step_history,
                   'Single Step Training and validation loss')
```

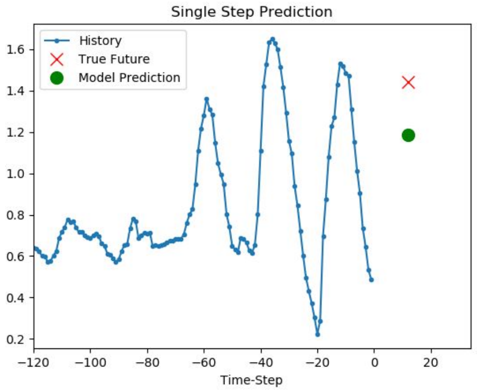


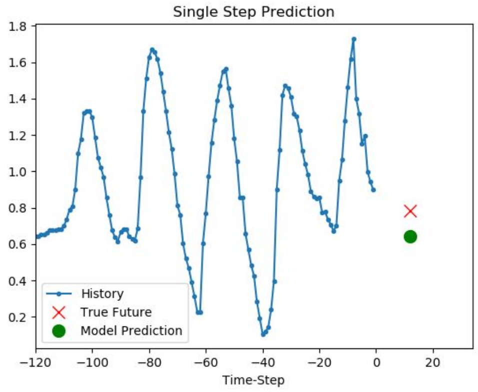
訓練データの損失関数は減少。評価データの損失関数にも特異な振る舞いはない。

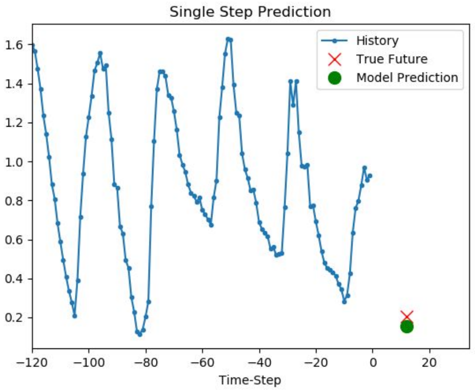
Lesson 4: 多変量の時系列予測 - プロット

```
for x, y in val_data_single.take(3):
  plot = show_plot([x[0][:, 1].numpy(), y[0].numpy(),
  single_step_model.predict(x)[0]], 12, 'Single Step Prediction')
  plot.show()
```

予測された気温の
3パターンが
表示される。







演習　コマンドライン：`$ python lesson4_multivariate_single_step.py utils.py`
jupyter notebook：# lesson4_multivariate_single_step の欄を Ctrl+Enter

Lesson 5: 多変量の時系列予測 - モデルの構築

気圧、気温、密度のデータからある時刻 (72点)の気温を予測する。

```
multi_step_model = tf.keras.models.Sequential()
multi_step_model.add(tf.keras.layers.LSTM(32,
                                          return_sequences=True,
                                          input_shape=x_train_multi.shape[-2:]))
multi_step_model.add(tf.keras.layers.LSTM(16, activation='relu'))
multi_step_model.add(tf.keras.layers.Dense(72))

multi_step_model.compile(optimizer=tf.keras.optimizers.RMSprop(clipvalue=1.0),
                         loss='mae')
```

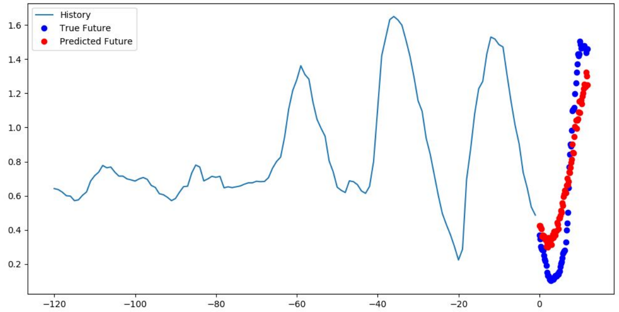
出力が32次元のLSTM層と、16次元のLSTM層を用意。72点を予測したいのでDense(72)とする。

Lesson 5: 多変量の時系列予測 - プロット

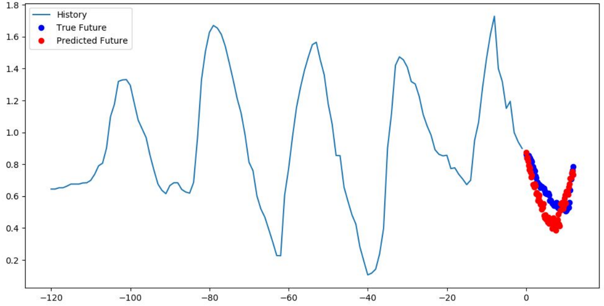
```
for x, y in val_data_multi.take(3):
multi_step_plot(x[0], y[0],
                multi_step_model.predict(x)[0])
```

予測された気温の3パターンが表示される。

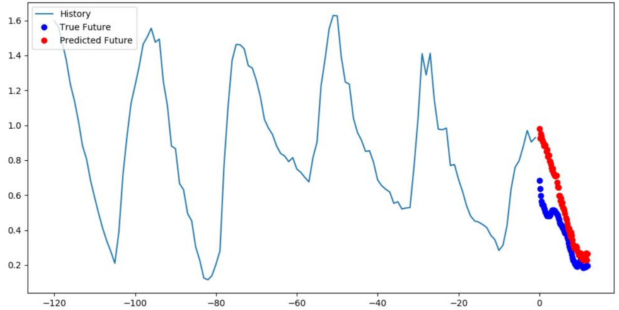
(1)



(2)



(3)



演習　コマンドライン：`$ python lesson5 multivariate multi-step.py utils.py`
jupyter notebook：# lesson5_multivariate_multi-step (1), (2) の欄を順に Ctrl+Enter
(コマンドラインで実行されている方は、最初のグラフが表示されたら閉じて下さい。先に進みます。)

参考文献 (書籍)

[1] Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, Chapter 15, 2nd edition, Aurélien Géron, O'Reilly.

[2] DEEP LEARNING with Python, Chapter 6, François Chollet, MANNING.

[3] 詳解 ディープラーニング TensorFlow·Kerasによる時系列データ処理、巣籠 悠輔、マイナビ出版。

[4] TensorFlowではじめるDeep Learning実装入門、新村 拓也、インプレス。

参考文献 (論文)

[1] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory”, Neural Computation 9, 1735-1780 (1997).

[2] H. Sak *et al*., “Long Short-Term Memory Based Recurrent Neural Network Architectures for Large Vocabulary Speech Recognition”, arXiv:1402.1128 (2014).

[3] W. Zaremba *et al*., “Recurrent Neural Network Regularization”, arXiv:1409.2329 (2014).