【第 9 章補足資料】

表 9.1　ExcelReg(書籍版 p.161 関連資料)

回帰統計	
重相関 R	0.864
重決定 R2	0.747
補正 R2	0.744
標準誤差	1.006
観測数	100

分散分析表

	自由度	変動	分散	分散比 F	有意 F
回帰	1	292.97	292.97	289.30	5.3E-31
残差	98	99.24	1.01		
合計	99	392.21			

	係数	標準誤差	t	P-値	下限 95%	上限 95%
定数	4.22	0.19	21.84	0.000	3.84	4.61
X	2.97	0.17	17.01	0.000	2.62	3.31

$H_0 : \beta_0 = 4$　　t= 1.16　採択

$H_1 : \beta_1 = 3$　　t= 0.18　採択

＜プログラム 9.1＞ SciReg

In [1]:

```
#%matplotlib inline
#import matplotlib as mpl
import matplotlib. pyplot as plt
import numpy as np
```

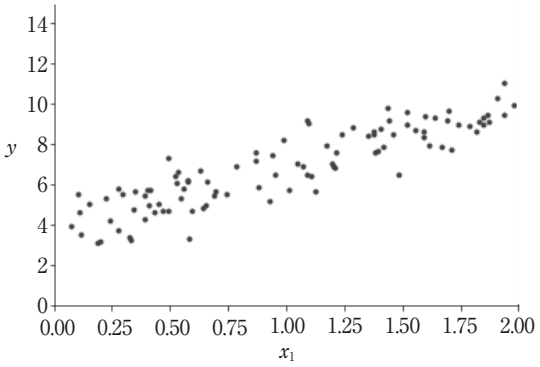
In [2]:

```
# テスト用の式の定義
X = 2 * np. random. rand (100, 1)
y = 4 + 3 * X + np. random. randn (100, 1)
```

In [3]:

```
#x,y の描画
plt. plot (X, y, "b.")
plt. xlabel ("$x_1$", fontsize=18)
plt. ylabel ("$y$", rotation=0, fontsize=18)
plt. axis ([0, 2, 0, 15])
plt. show ()
```

【第 9 章補足資料】



In [4]:

```
X_b = np. c_[np. ones ((100, 1)), X] # add x0 = 1 to each instance
theta_best = np. linalg. inv (X_b. T. dot (X_b)). dot (X_b. T). dot (y)
```

In [5]:

```
theta_best
```

Out[5]:

```
array ([[3. 82373444],
        [3. 05809045]])
```

In [6]:

```
X_new = np. array ([[0], [2]])
X_new_b = np. c_[ np. ones ((2, 1)), X_new] # add x0 = 1 to each instance
y_predict = X_new_b. dot (theta_best)
y_predict
```

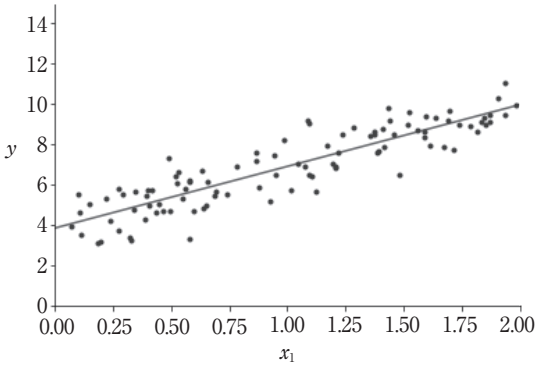
Out[6]:

```
array ([[3. 82373444],
        [9. 93991533]])
```

In [7]:

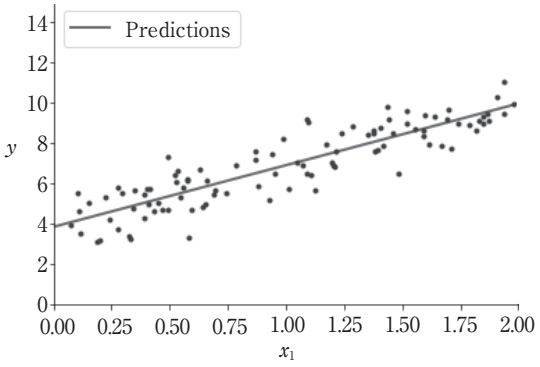
```
plt. plot (X_new, y_predict, "r-")
plt. plot (X, y, "b.")
plt. axis ([0, 2, 0, 15])
plt. show ()
```

【第 9 章補足資料】



In [8]:

```
plt. plot (X_new, y_predict, "r-", linewidth=2, label="Predictions")
plt. plot (X, y, "b.")
plt. xlabel ("$x_1$", fontsize=18)
plt. ylabel ("$y$", rotation=0, fontsize=18)
plt. legend (loc="upper left", fontsize=14)
plt. axis ([0, 2, 0, 15])
plt. show ()
```



【第 9 章補足資料】

In [9]:

```
from sklearn. linear_model import LinearRegression

lin_reg = LinearRegression ()
lin_reg. fit (X, y)
lin_reg. intercept_, lin_reg. coef_
```

Out[9]:

```
(array ([3. 82373444]), array([[3. 05809045]]))
```

In [10]:

```
lin_reg. predict (X_new)
```

Out[10]:

```
array ([[3. 82373444],
        [9. 93991533]])
```

The LinearRegression class is based on the scipy.linalg.lstsq() function (the name stands for "leastsquares"), which you could call directly:

In [11]:

```
theta_best_svd, residuals, rank, s = np. linalg. lstsq (X_b, y, rcond=1e-6)
theta_best_svd
```

Out[11]:

```
array ([[3. 82373444],
        [3. 05809045]])
```

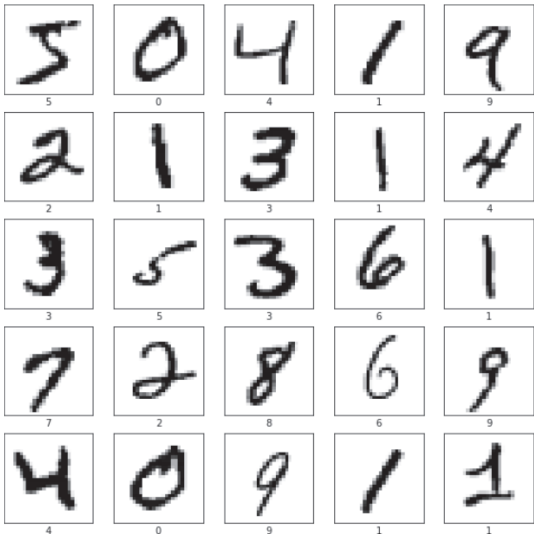
【第 9 章補足資料】

図 9.2 補足（書籍版 p.163 参照）
＜プログラム 9.2＞ 8 の判定

①プログラム・インポート

```
# 関連ライブラリのインポート
import tensorflow as tf
from tensorflow import keras

import numpy as np
import matplotlib.pyplot as plt
import cv2
from PIL import Image



#MNIST 読込
mnist = keras.datasets.mnist
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()



# データの事前処理
class_names = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']
train_images = train_images / 255.0
test_images = test_images / 255.0



# 訓練用データセットの表示 (最初の 25 枚)

plt.figure(figsize=(10,10))
for i in range(25):
  plt.subplot(5,5,i+1)
  plt.xticks([])
  plt.yticks([])
  plt.grid(False)
  plt.imshow(train_images[i], cmap=plt.cm.binary)
  plt.xlabel(class_names[train_labels[i]])
plt.show()
```

【第 9 章補足資料】



②訓練，③テスト，④予測のモデル

②③の予測と④の‘本番’の予測がある．またコンパイル中に，クロス・エントロピー(第 5 章)がロジスティック関数(後述)の最適化基準に採用されていることに注目．

```
# モデルの構築
model = keras.Sequential([
  keras.layers.Flatten(input_shape=(28, 28)),
  keras.layers.Dense(128, activation='relu'),
  keras.layers.Dense(10, activation='softmax')
])

# モデルのコンパイル
model.compile(optimizer='adam',
      loss='sparse_categorical_crossentropy',
      metrics=['accuracy'])

# モデルの訓練
model.fit(train_images, train_labels, epochs=5)
```

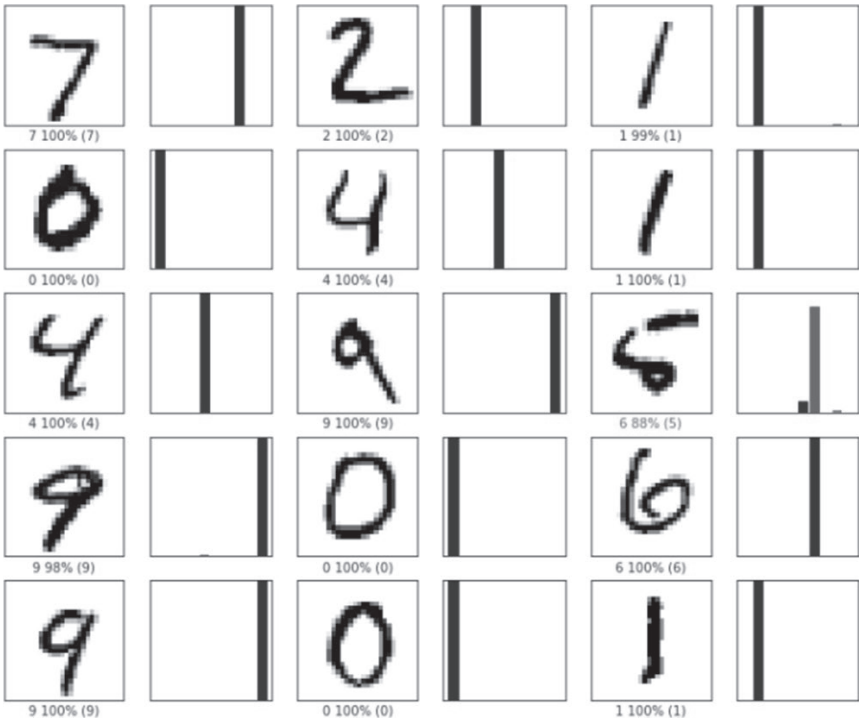
【第 9 章補足資料】

```
# 予測
predictions = model.predict(test_images)

# 予測結果の表示

def plot_image(i, predictions_array, true_label, img):
  predictions_array, true_label, img = predictions_array[i] , true_label[i] ,
  img[i]
  plt.grid(False)
  plt.xticks([])
  plt.yticks([])

  plt.imshow(img, cmap=plt.cm.binary)

  predicted_label = np.argmax(predictions_array)
  if predicted_label == true_label:
    color = 'blue'
  else:
    color = 'red'

  plt.xlabel("{} {:2.0f}% ({})".format(class_names[predicted_label],
                       100*np.max(predictions_array),
                       class_names[true_label]),
                       color=color)


def plot_value_array(i, predictions_array, true_label):
  predictions_array, true_label = predictions_array[i], true_label[i]
  plt.grid(False)
  plt.xticks([])
  plt.yticks([])
  thisplot = plt.bar(range(10), predictions_array, color="#777777")
  plt.ylim([0, 1])
  predicted_label = np.argmax(predictions_array)
```

【第 9 章補足資料】

```
  thisplot[predicted_label].set_color('red')
  thisplot[true_label].set_color('blue')


#X 個のテスト画像，予測されたラベル，正解ラベルを表示します．
# 正しい予測は青で，間違った予測は赤で表示しています．
num_rows = 5
num_cols = 3
num_images = num_rows*num_cols
plt.figure(figsize=(2*2*num_cols, 2*num_rows))
for i in range(num_images):
  plt.subplot(num_rows, 2*num_cols, 2*i+1)
  plot_image(i, predictions, test_labels, test_images)
  plt.subplot(num_rows, 2*num_cols, 2*i+2)
  plot_value_array(i, predictions, test_labels)
plt.show()
```

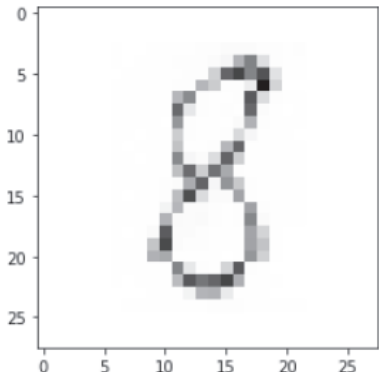
【第 9 章補足資料】



※柱の位置は左⇒右と 10 通り（0, 1, 2, ……, 9），高さは確率 0.0～1.0 を表す．ほとんどが高さ 1.0（100%）で完全予測だが，中段右の「5」は，予測結果では 88% の確率で「6」と予測している．訓練を繰り返すことで，正解率は上がっていく．

【第 9 章補足資料】

```
#8 の画像の予測



# 画像の読込
test_labels=[8]  # 正解を設定
im =
np.array(Image.open(r'C:¥SIP¥20160611_stocks_hop_on¥ap2¥source¥___matsu
bara¥imageX.bmp').convert('L'))
img=1-(im/255)

plt.imshow(img, cmap=plt.cm.binary)
```



※読み込まれた画像

⑤予測の表示　結果は‘8’（56.9%），‘5’（17.7%）‘6’（13.3%）となっている．

```
# 予測

# 画像を 1 枚だけのバッチのメンバーにする
img = (np.expand_dims(img,0))
predictions_single = model.predict(img)

for x in range(0,10):
  print("{} -> {:.1%}".format(x,predictions_single[0][x]))
```

【第 9 章補足資料】

```
predictions_single
# 結果表示
plot_value_array(0, predictions_single, test_labels)
_ = plt.xticks(range(10), class_names, rotation=45)
```

```
0 -> 1.7%
1 -> 1.2%
2 -> 2.4%
3 -> 4.7%
4 -> 0.3%
5 -> 17.7%
6 -> 13.7%
7 -> 0.5%
8 -> 56.9%
9 -> 1.0%
```

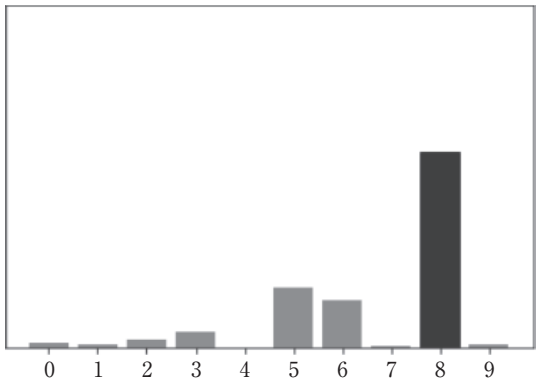


図 9.2　MNIST データ
［By Josef Steppan -, CC BY-SA 4.0,
https://commons.wikimedia.org/w/index.php?curid=64810040］

【第 9 章補足資料】

（書籍版 p.165 参照）

＜プログラム 9.3＞ IrisSVM

おなじみのアイリスデータ（第 6 章）を分離してみよう

In [1]:
```
# 関連ライブラリインポート
import numpy as np
import matplotlib as mpl
import matplotlib. pyplot as plt
from sklearn. svm import SVC
from sklearn import datasets
```

In [2]:
```
# モデルの作成
iris = datasets. load_iris()
X = iris["data"][:, (2, 3)] # petal length, petal width
y = iris["target"]

setosa_or_versicolor = (y == 0) | (y == 1)
X = X[setosa_or_versicolor]
y = y[setosa_or_versicolor]

# SVM Classifier model
svm_clf = SVC(kernel= "linear", C= float("inf"))
svm_clf. fit(X, y)
```

Out[2]:
```
SVC(C=inf, kernel='linear')
```

In [8]:
```
# Bad models
x0 = np. linspace(0, 5.5, 200)
pred_1 = 5*x0 - 20
pred_2 = x0 - 1.8
pred_3 = 0.1 * x0 + 0.5
def plot_svc_decision_boundary(svm_clf, xmin, xmax):
    #svm のマージンの描画
    w = svm_clf. coef_[0]
    b = svm_clf. intercept_[0]

    # At the decision boundary, w0*x0 + w1*x1 + b = 0
    # => x1 = -w0/w1 * x0 - b/w1
```

```
    x0 = np. linspace(xmin, xmax, 200)
    decision_boundary = -w[0]/ w[1] * x0 - b/w[1]
    margin = 1/w[1]
    gutter_up = decision_boundary + margin
    gutter_down = decision_boundary - margin

    svs = svm_clf. support_vectors_
    plt. scatter(svs[:, 0], svs[:, 1], s=180, facecolors='#FFAAAA')
    plt. plot(x0, decision_boundary, "k-", linewidth=2)
    plt. plot(x0, gutter_up, "k--", linewidth=2)
    plt. plot(x0, gutter_down, "k--", linewidth=2)

fig, axes = plt. subplots(ncols=2, figsize=(10,2.7), sharey=True)

# 直線による分類の例の描画
plt. sca(axes[0])
plt. plot(x0, pred_1, "g--", linewidth=2)
plt. plot(x0, pred_2, "m-", linewidth=2)
plt. plot(x0, pred_3, "r-", linewidth=2)
plt. plot(X[:, 0][y==1], X[:, 1][y==1], "bs", label="Iris versicolor")
plt. plot(X[:, 0][y==0], X[:, 1][y==0], "yo", label="Iris setosa")
plt. xlabel("Petal length", fontsize= 14)
plt. ylabel("Petal width", fontsize=14)
plt. legend(loc= "upper left", fontsize=14)
plt. axis([0, 5.5, 0, 2])

#svm により分類した例の描画
plt. sca(axes[1])
plot_svc_decision_boundary(svm_clf, 0, 5.5)
plt. plot(X[:, 0][y==1], X[:, 1][y==1], "bs")
plt. plot(X[:, 0][y==0], X[:, 1][y==0], "yo")
plt. xlabel("Petal length", fontsize=14)
plt. axis([0, 5.5, 0, 2])

plt. show()
```

【第 9 章補足資料】

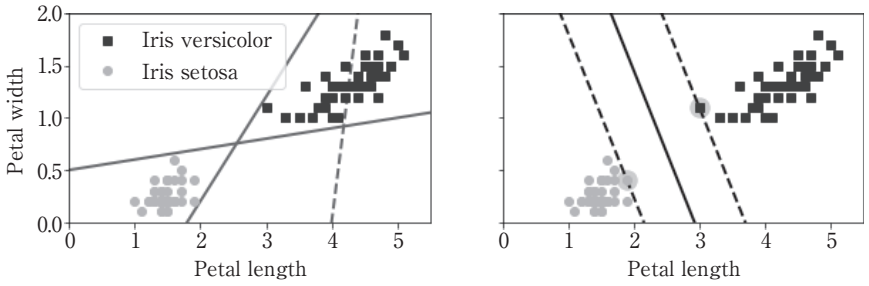


図 9.3　アイリス・データの SVM：ベルシカラー対セトーサ（書籍版 p.165 参照）
横軸＝花弁長，縦軸＝花弁幅．右図は最適化例，左図は良くない教育例