分类

本节代码来自: 书籍代码 https://github.com/ageron/handson-ml3 推荐自学

1.1 数据准备

In [1]:
```
import warnings
warnings.filterwarnings("ignore")

import matplotlib.pyplot as plt

plt.rc('font', size=14)
plt.rc('axes', labelsize=14, titlesize=14)
plt.rc('legend', fontsize=14)
plt.rc('xtick', labelsize=10)
plt.rc('ytick', labelsize=10)
```

In [2]:
```
from pathlib import Path

IMAGES_PATH = Path() / "images"
IMAGES_PATH.mkdir(parents=True, exist_ok=True)

def save_fig(fig_id, tight_layout=True, fig_extension="png", resolution=300):
    path = IMAGES_PATH / f"{fig_id}.{fig_extension}"
    if tight_layout:
        plt.tight_layout()
    plt.savefig(path, format=fig_extension, dpi=resolution)
```

In [3]:
```
from sklearn.datasets import fetch_openml
mnist = fetch_openml('mnist_784', as_frame=False)
mnist.keys()
```

Out[3]:
```
dict_keys(['data', 'target', 'frame', 'categories', 'feature_names', 'target_names', 'DESCR', 'details', 'url'])
```

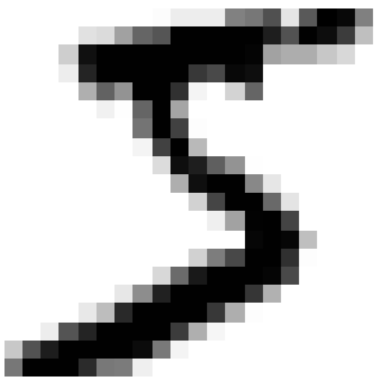
In [4]:
```
X, y = mnist["data"], mnist["target"]
print(X.shape)
print(y.shape)
```

```
(70000, 784)
(70000,)
```

In [5]:
```
import matplotlib.pyplot as plt

def plot_digit(image_data):
    image = image_data.reshape(28, 28)
    plt.imshow(image, cmap="binary")
    plt.axis("off")

some_digit = X[0]
plot_digit(some_digit)
save_fig("some_digit_plot")  # extra code
plt.show()
```



In [6]:
```
# extra code – this cell generates and saves Figure 3–2
plt.figure(figsize=(9, 9))
for idx, image_data in enumerate(X[:100]):
    plt.subplot(10, 10, idx + 1)
    plot_digit(image_data)
plt.subplots_adjust(wspace=0, hspace=0)
save_fig("more_digits_plot", tight_layout=False)
plt.show()
```

In [7]:
```
#MNIST数据集已经分成训练集（前6万张图片）和测试集（最后1万张图片）
X_train, X_test, y_train, y_test = X[:60000], X[60000:], y[:60000], y[60000:]
```

1.2 训练一个二分类器

现在先简化问题，只尝试识别一个数字，比如数字5。那么这个"数字5检测器"就是一个二元分类器的示例，它只能区分两个类别： 5和非5。先为此分类任务创建目标向量

In [8]:
```
y_train_5 = (y_train == '5')  # True for all 5s, False for all other digits
y_test_5 = (y_test == '5')
```

```
print(y_train[:6])
print(y_train_5[:6])
```

```
['5' '0' '4' '1' '9' '2']
[ True False False False False False]
```

In [9]:
```
from sklearn.linear_model import SGDClassifier

sgd_clf = SGDClassifier(random_state=2023)
sgd_clf.fit(X_train, y_train_5)
```

Out[9]:
```
▾ SGDClassifier
SGDClassifier(random_state=2023)
```

In [10]:
```
sgd_clf.predict([some_digit])
```

Out[10]:
```
array([ True])
```

1.3 性能度量

1.3.1 使用交叉验证测量准确率

In [11]:
```
from sklearn.model_selection import cross_val_score

cross_val_score(sgd_clf, X_train, y_train_5, cv=3, scoring="accuracy")
```

Out[11]:
```
array([0.9587 , 0.9572 , 0.96035])
```

DummyClassifier是一种使用简单规则进行预测的分类器。

这个分类器作为与其他(真实的)分类器进行比较的简单基线非常有用。不要用它来解决真正的问题。

- any() 函数
  - 用于判断内容是否全部为空，全部为空则返回 False，否则返回 True.
  - 空元素包括：0、空、FALSE

dummy分类器将每张图都分类成“非5”

In [12]:
```
from sklearn.dummy import DummyClassifier

dummy_clf = DummyClassifier()
dummy_clf.fit(X_train, y_train_5)
print(any(dummy_clf.predict(X_train)))
```

```
False
```

In [13]:
```
cross_val_score(dummy_clf, X_train, y_train_5, cv=3, scoring="accuracy")
```

Out[13]:
```
array([0.90965, 0.90965, 0.90965])
```

1.3.2 混淆矩阵

In [14]:
```
from sklearn.model_selection import cross_val_predict
y_train_pred = cross_val_predict(sgd_clf, X_train, y_train_5, cv=3)
```

In [15]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_train_5, y_train_pred)
cm
```

Out[15]:
```
array([[53205,  1374],
       [ 1101,  4320]], dtype=int64)
```

In [16]:
```
from sklearn.metrics import precision_score, recall_score

precision_score(y_train_5, y_train_pred)  # == 4320 / (1374+4320)
# extra code – this cell also computes the precision: TP / (FP + TP)
# cm[1, 1] / (cm[0, 1] + cm[1, 1])
```

Out[16]:
```
0.7586933614330874
```

In [17]:
```
recall_score(y_train_5, y_train_pred)  # == 4320 / (1101+4320)
# extra code – this cell also computes the recall: TP / (FN + TP)
# cm[1, 1] / (cm[1, 0] + cm[1, 1])
```

Out[17]:
```
0.7969009407858328
```

In [18]:
```
from sklearn.metrics import f1_score
f1_score(y_train_5, y_train_pred)
```

```
# extra code – this cell also computes the f1 score
# cm[1, 1] / (cm[1, 1] + (cm[1, 0] + cm[0, 1]) / 2)
```

Out[18]: 0.777327935222672

In [19]:
```
y_scores = cross_val_predict(sgd_clf, X_train, y_train_5, cv=3,
                             method="decision_function")
from sklearn.metrics import precision_recall_curve

precisions, recalls, thresholds = precision_recall_curve(y_train_5, y_scores)
```

1.4 多分类

SVM不能很好地扩展到大型数据集，因此让我们仅在前2,000个实例上进行训练，否则将需要很长时间才能运行：

In [20]:
```
from sklearn.svm import SVC
svm_clf = SVC(random_state=2023)
svm_clf.fit(X_train[:2000], y_train[:2000])  # y_train, not y_train_5
```

Out[20]:
```
▾ SVC
SVC(random_state=2023)
```

In [21]:
```
svm_clf.predict([some_digit])
```

Out[21]: array(['5'], dtype=object)

In [22]:
```
some_digit_scores = svm_clf.decision_function([some_digit])
some_digit_scores
```

Out[22]:
```
array([[ 3.79297828,  0.72949369,  6.06184129,  8.29800527, -0.29383983,
         9.30157597,  1.74723215,  2.77365456,  7.20601456,  4.82245092]])
```

In [23]:
```
# extra code – shows how to get all 45 OvO scores if needed
svm_clf.decision_function_shape = "ovo"
some_digit_scores_ovo = svm_clf.decision_function([some_digit])
print(some_digit_scores_ovo.shape)
some_digit_scores_ovo.round(2)
```

```
(1, 45)
```

Out[23]:
```
array([[ 0.11, -0.21, -0.97,  0.51, -1.01,  0.19,  0.09, -0.31, -0.04,
        -0.45, -1.28,  0.25, -1.01, -0.13, -0.32, -0.9 , -0.36, -0.93,
         0.79, -1.  ,  0.45,  0.24, -0.24,  0.25,  1.54, -0.77,  1.11,
         1.13,  1.04,  1.2 , -1.42, -0.53, -0.45, -0.99, -0.95,  1.21,
         1.  ,  1.  ,  1.08, -0.02, -0.67, -0.14, -0.3 , -0.13,  0.25]])
```

In [24]:
```python
sgd_clf = SGDClassifier(random_state=2023)
sgd_clf.fit(X_train, y_train)
sgd_clf.predict([some_digit])
```

Out[24]:
```
array(['5'], dtype='<U1')
```

In [25]:
```python
sgd_clf.decision_function([some_digit]).round()
```

Out[25]:
```
array([[-10971., -30760.,  -8162.,    957., -16908.,   1129., -21757.,
        -21891., -10384., -15867.]])
```

Warning: 以下两个单元格可能需要几分钟才能运行：

In [26]:
```python
cross_val_score(sgd_clf, X_train, y_train, cv=3, scoring="accuracy")
```

Out[26]:
```
array([0.86135, 0.8661 , 0.88225])
```

In [27]:
```python
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train.astype("float64"))
cross_val_score(sgd_clf, X_train_scaled, y_train, cv=3, scoring="accuracy")
```

Out[27]:
```
array([0.89745, 0.89205, 0.90255])
```

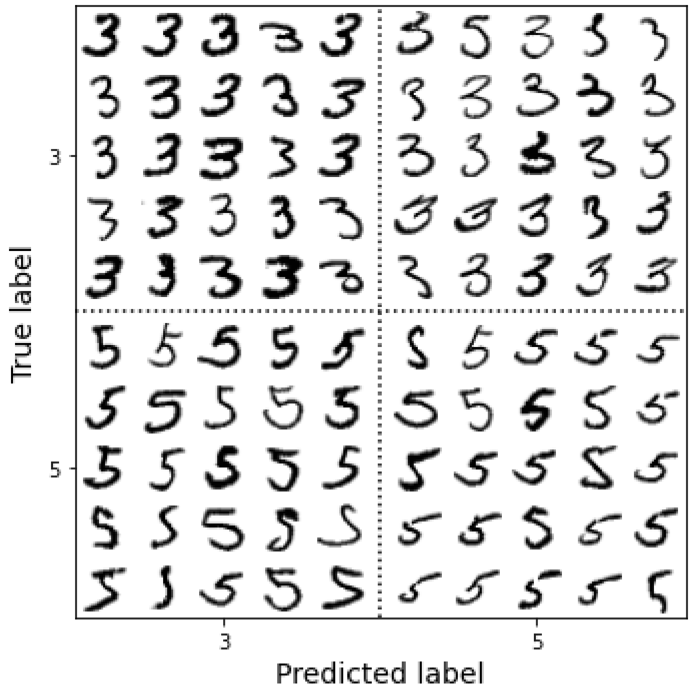
1.5 误差分析

In [ ]:
```python
from sklearn.metrics import ConfusionMatrixDisplay

y_train_pred = cross_val_predict(sgd_clf, X_train_scaled, y_train, cv=3)
plt.rc('font', size=9)  # extra code – make the text smaller
ConfusionMatrixDisplay.from_predictions(y_train, y_train_pred)
save_fig("confusion_matrix_plot")
plt.show()
```

In [30]:
```
cl_a, cl_b = '3', '5'
X_aa = X_train[(y_train == cl_a) & (y_train_pred == cl_a)]
X_ab = X_train[(y_train == cl_a) & (y_train_pred == cl_b)]
X_ba = X_train[(y_train == cl_b) & (y_train_pred == cl_a)]
X_bb = X_train[(y_train == cl_b) & (y_train_pred == cl_b)]
```

In [31]:
```
size = 5
pad = 0.2
plt.figure(figsize=(size, size))
for images, (label_col, label_row) in [(X_ba, (0, 0)), (X_bb, (1, 0)),
                                       (X_aa, (0, 1)), (X_ab, (1, 1))]:
    for idx, image_data in enumerate(images[:size*size]):
        x = idx % size + label_col * (size + pad)
        y = idx // size + label_row * (size + pad)
        plt.imshow(image_data.reshape(28, 28), cmap="binary",
                   extent=(x, x + 1, y, y + 1))
plt.xticks([size / 2, size + pad + size / 2], [str(cl_a), str(cl_b)])
plt.yticks([size / 2, size + pad + size / 2], [str(cl_b), str(cl_a)])
plt.plot([size + pad / 2, size + pad / 2], [0, 2 * size + pad], "k:")
plt.plot([0, 2 * size + pad], [size + pad / 2, size + pad / 2], "k:")
plt.axis([0, 2 * size + pad, 0, 2 * size + pad])
plt.xlabel("Predicted label")
plt.ylabel("True label")
save_fig("error_analysis_digits_plot")
plt.show()
```



In [32]:
```
X_aa.shape
```

Out[32]:
```
(5226, 784)
```