یادگیری عمیق

درس ۱۰

مروری بر شبکه‌های عصبی مصنوعی (۲)

An Overview on Artificial Neural Networks (2)

کاظم فولادی قلعه

دانشکده مهندسی، دانشکدگان فارابی

دانشگاه تهران

http://courses.fouladi.ir/deep

یادگیری عمیق

مروری بر شبکه‌های عصبی مصنوعی

۵

طبقه‌بندی چندکلاسی

مثال: طبقه‌بندی خطوط خبری

طبقه‌بندی خطوط خبری

یک مثال برای طبقه‌بندی چندطبقه‌ای

CLASSIFYING NEWSWIRES: A MULTICLASS CLASSIFICATION EXAMPLE

طبقه‌بندی برای حالتی که بیش از دو طبقه/کلاس داریم.

در این مثال:
هدف، یادگیری طبقه‌ی خطوط خبری رویترز در ۴۶ موضوع متمایز است.

Because you have many classes, this problem is an instance of *multiclass classification*; and because each data point should be classified into only one category, the problem is more specifically an instance of *single-label, multiclass classification*.
If each data point could belong to multiple categories (in this case, topics), you'd be facing a *multilabel, multiclass classification* problem.

طبقه‌بندی خطوط خبری

مجموعه‌داده‌ی رویترز

THE REUTERS DATASET

Reuters Dataset

a set of short newswires and their topics, published by Reuters in 1986. It's a simple, widely used toy dataset for text classification. There are 46 different topics; some topics are more represented than others, but each topic has at least 10 examples in the training set.

طبقه‌بندی خطوط خبری

خواندن مجموعه‌داده‌ی Reuters

Listing 4.11 Loading the Reuters dataset

```
from tensorflow.keras.datasets import reuters
(train_data, train_labels), (test_data, test_labels) = reuters.load_data(
    num_words=10000)
```

آرگومان num_words=10000 به معنی این است که ۱۰۰۰۰ کلمه‌ی فراوان‌تر در داده‌های آموزشی را نگه می‌داریم.
کلمات نادرتر کنار گذاشته می‌شوند ⇐ می‌توانیم با بردارهایی با اندازه‌ی ثابت کار کنیم.

متغیرهای train_data و test_data لیست‌هایی از اخبار هستند: هر خبر لیستی از اندیس کلمات است.
train_labels و test_labels لیست‌هایی از اعداد 0 تا 45 (اندیس موضوع) هستند.

```
>>> len(train_data)
8982
>>> len(test_data)
2246
```

```
>>> train_data[10]
[1, 245, 273, 207, 156, 53, 74, 160, 26, 14, 46, 296,
26, 39, 74, 2979,3554, 14, 46, 4689, 4329, 86, 61,
3499, 4795, 14, 61, 451, 4329, 17, 12]
```

```
>>> train_labels[10]
3
```

طبقه‌بندی خطوط خبری

استخراج متن خبرها

Listing 4.12 Decoding newswires back to text

```
word_index = reuters.get_word_index()
reverse_word_index = dict(
    [(value, key) for (key, value) in word_index.items()])
decoded_newswire = " ".join(
    [reverse_word_index.get(i - 3, "?") for i in     train_data[0]])
```

Note that the indices are offset by 3 because 0, 1, and 2 are reserved indices for “padding,” “start of sequence,” and “unknown.”

طبقه‌بندی خطوط خبری

آماده‌سازی داده‌ها

PREPARING THE DATA

برداری‌سازی داده‌ها:

Listing 4.13 Encoding the input data

```
x_train = vectorize_sequences(train_data)
x_test = vectorize_sequences(test_data)
```

Vectorized training data

Vectorized test data

طبقه‌بندی خطوط خبری

آماده‌سازی داده‌ها

PREPARING THE DATA

برداری‌سازی برچسب‌ها:

راه‌های برداری‌سازی برچسب‌ها		
کدگذاری عددی *Integer Encoding*	می‌توان لیست برچسب‌ها را به‌صورت تانسوری از اعداد صحیح تبدیل کرد.	
کدگذاری «تک–داغ» *One-Hot Encoding*	**کدگذاری دسته‌ای** *Categorical Encoding*	لیست برچسب‌ها را به بردارهایی از 0ها و 1 ها تبدیل می‌کنیم (کدگذاری تک–داغ): [همه‌جا صفر، غیر از اندیس برچسب حاضر در لیست]

طبقه‌بندی خطوط خبری

آماده‌سازی داده‌ها

PREPARING THE DATA

برداری‌سازی برچسب‌ها:

Listing 4.14 Encoding the labels

```
def to_one_hot(labels, dimension=46):
    results = np.zeros((len(labels), dimension))
    for i, label in enumerate(labels):
        results[i, label] = 1.
    return results
y_train = to_one_hot(train_labels)
y_test = to_one_hot(test_labels)
```

Vectorized training labels

Vectorized test labels

Note that there is a built-in way to do this in **Keras**,
which you've already seen in action in the MNIST example:

```
from tensorflow.keras.utils import to_categorical
y_train = to_categorical(train_labels)
y_test = to_categorical(test_labels)
```

طبقه‌بندی خطوط خبری

ساخت شبکه

BUILDING THE NETWORK

مسئله‌ی طبقه‌بندی موضوع‌ها، بسیار شبیه به مسئله‌ی طبقه‌بندی نقد فیلم‌هاست:
در هر دو سعی می‌کنیم تکه‌های کوتاهی از متن را طبقه‌بندی کنیم.

اما

در مسئله‌ی طبقه‌بندی موضوع‌ها، قید جدیدی وجود دارد:
تعداد کلاس‌های خروجی از ۲ به ۴۶ تغییر کرده است: بعد فضای خروجی بسیار بزرگ‌تر شده است.

در پشته‌ی لایه‌های Dense هر لایه فقط به اطلاعات حاضر در خروجی لایه‌ی قبلی دسترسی دارد.
اگر یک لایه برخی اطلاعات مربوط به مسئله‌ی طبقه‌بندی را حذف کند،
این اطلاعات هرگز نمی‌توانند توسط لایه‌های بعدی بازسازی شوند:
هر لایه می‌تواند به‌صورت بالقوه یک گلوگاه اطلاعاتی باشد.

در مثال قبلی، از لایه‌های میانی ۱۶-بعدی استفاده کردیم،
اما فضای ۱۶-بعدی برای یادگیری ۴۶ کلاس مختلف ممکن است بسیار محدود باشد:
برای همین از لایه‌های ۶۴ واحدی استفاده می‌کنیم.

طبقه‌بندی خطوط خبری

تعریف مدل شبکه

Listing 4.15 Model definition

```
model = keras.Sequential([
    layers.Dense(64, activation="relu"),
    layers.Dense(64, activation="relu"),
    layers.Dense(46, activation="softmax")
])
```

شبکه با یک لایه‌ی Dense با اندازه‌ی ۴۶ خاتمه یافته است.
یعنی: برای هر نمونه‌ی ورودی، خروجی شبکه یک بردار ۴۶–بعدی است.
هر درایه در این بردار (هر بعد این بردار) یک کلاس خروجی متفاوت را کدگذاری می‌کند.

لایه‌ی آخر از تابع فعالیت softmax استفاده می‌کند.
یعنی: این شبکه یک توزیع احتمال بر روی ۴۶ کلاس خروجی متفاوت برمی‌گرداند.
(output[i] احتمال تعلق نمونه‌ی ورودی به کلاس i را نشان می‌دهد.)
مجموع این ۴۶ امتیاز، برابر با یک است.

طبقه‌بندی خطوط خبری

کامپایل کردن مدل

COMPILING THE MODEL

با یک مسئله‌ی طبقه‌بندی چندکلاسی مواجه هستیم و خروجی شبکه یک احتمال است
⇐ بهترین انتخاب برای تابع اتلاف `categorical_crossentropy loss` است.

Listing 4.16 Compiling the model

```
model.compile(optimizer="rmsprop",
              loss="categorical_crossentropy",
              metrics=["accuracy"])
```

$$CCE = -\frac{1}{N}\sum_{i=0}^{N}\sum_{j=0}^{J} y_j \cdot log(\hat{y}_j) + (1 - y_j) \cdot log(1 - \hat{y}_j)$$

طبقه‌بندی خطوط خبری

اعتبارسنجی روی‌کرد

VALIDATING THE APPROACH

برای نظارت بر دقت مدل «بر روی داده‌هایی که هرگز تاکنون دیده نشده‌اند» در حین آموزش،
باید یک **مجموعه‌ی اعتبارسنجی** ایجاد کنیم.
برای این منظور، ۱۰۰۰ نمونه از داده‌های آموزشی اصلی جدا می‌کنیم.

Listing 4.17 Setting aside a validation set

```
x_val = x_train[:1000]
partial_x_train = x_train[1000:]
y_val = y_train[:1000]
partial_y_train = y_train[1000:]
```

طبقه‌بندی خطوط خبری

آموزش مدل

TRAINING THE MODEL

حال باید مدل را آموزش بدهیم: ۲۰ اپک (epoch) = ۲۰ تکرار بر روی همه‌ی نمونه‌ها در داده‌های آموزشی
با ریزدسته‌های دارای ۵۱۲ نمونه.
همزمان، بر روی مقادیر **اتلاف** و **دقت** بر روی ۱۰۰۰ داده‌ی اعتبارسنجی جدا شده نظارت می‌کنیم.

Listing 4.18 Training the model

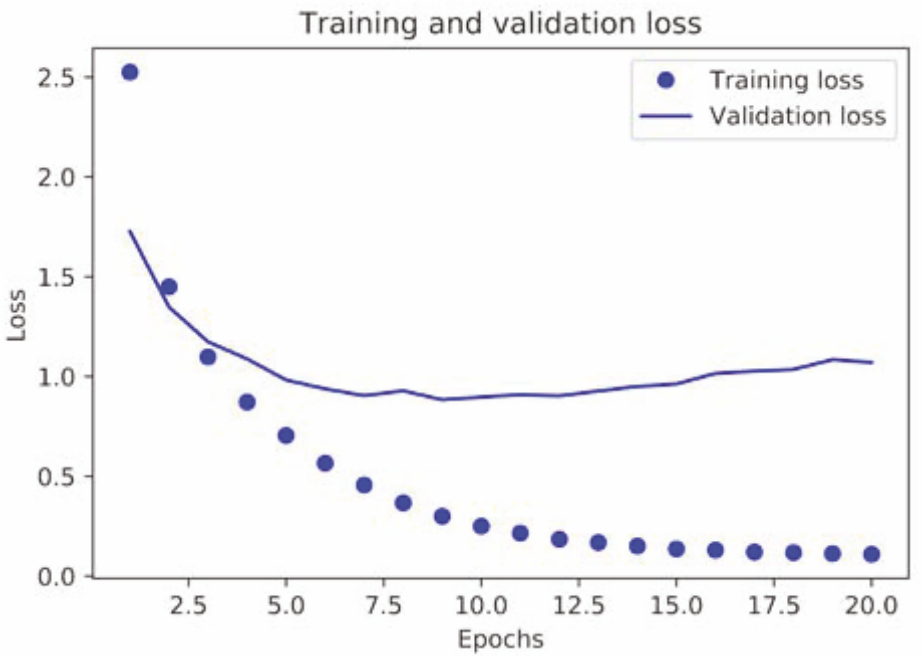
```
history = model.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val))
```

طبقه‌بندی خطوط خبری

رسم نمودار اتلاف

Listing 4.19 Plotting the training and validation loss

```
loss = history.history["loss"]
val_loss = history.history["val_loss"]
epochs = range(1, len(loss) + 1)
plt.plot(epochs, loss, "bo", label="Training loss")
plt.plot(epochs, val_loss, "b", label="Validation loss")
plt.title("Training and validation loss")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend()
plt.show()
```



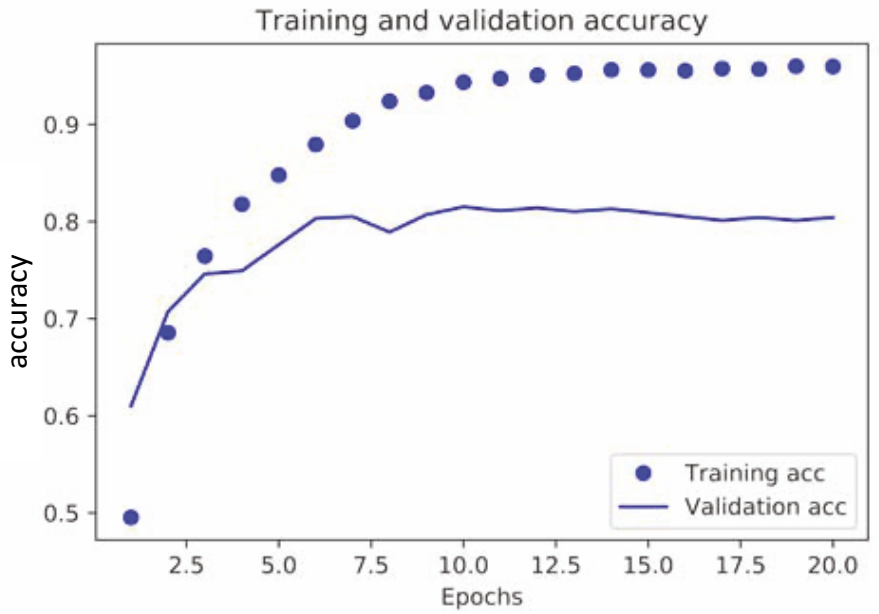
طبقه‌بندی خطوط خبری

رسم نمودار دقت

Listing 4.20 Plotting the training and validation accuracy

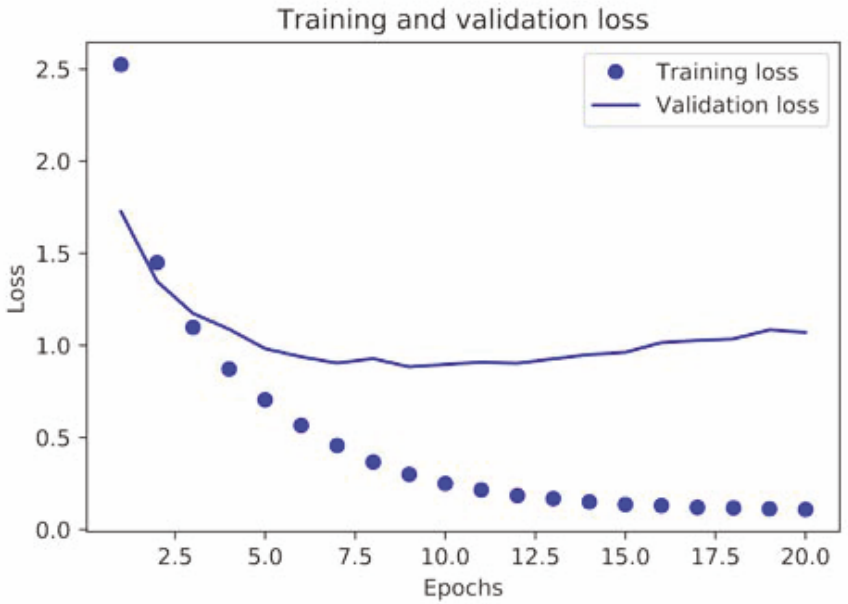
```
plt.clf()
acc = history.history["accuracy"]
val_acc = history.history["val_accuracy"]
plt.plot(epochs, acc, "bo", label="Training accuracy")
plt.plot(epochs, val_acc, "b", label="Validation accuracy")
plt.title("Training and validation accuracy")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.legend()
plt.show()
```

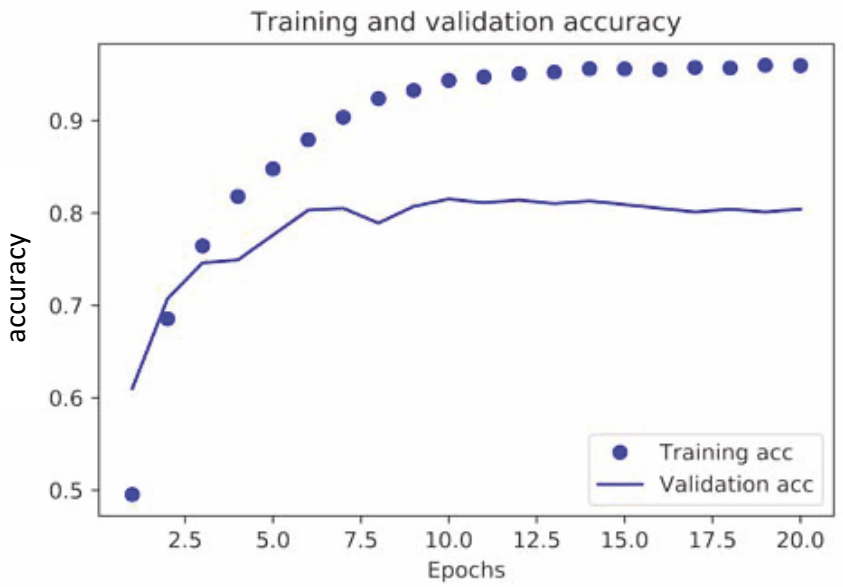
Clears the figure



طبقه‌بندی خطوط خبری

تحلیل نمودارهای اتلاف و دقت





اتلاف روی داده‌های آموزشی با هر اپک کاهش می‌یابد.
دقت بر روی داده‌های آموزشی بر روی هر اپک افزایش می‌یابد.

اما برای داده‌های اعتبارسنجی این‌گونه نیست: در اپک **نهم** به قله می‌رسد. ⇐ بیش‌برازش

طبقه‌بندی خطوط خبری

جلوگیری از بیش‌برازش

در این مورد، برای جلوگیری از بیش‌برازش، می‌توانیم آموزش را پس از ۹ اپک متوقف کنیم.

Listing 4.21 Retraining a model from scratch

```
model = keras.Sequential([
    layers.Dense(64, activation="relu"),
    layers.Dense(64, activation="relu"),
    layers.Dense(46, activation="softmax")
])
model.compile(optimizer="rmsprop",
              loss="categorical_crossentropy",
              metrics=["accuracy"])
model.fit(x_train,
          y_train,
          epochs=9,
          batch_size=512)
results = model.evaluate(x_test, y_test)
```

```
>>> results
[0.9565213431445807, 0.79697239536954589]
```

This approach reaches an accuracy of ~80%. With a balanced binary classification problem, the accuracy reached by a purely random classifier would be 50%.

This approach reaches an accuracy of ~80%. With a balanced binary classification problem, the accuracy reached by a purely random classifier would be 50%. But in this case, we have 46 classes, and they may not be equally represented. What would be the accuracy of a random baseline? We could try quickly implementing one to check this empirically:

```
>>> import copy
>>> test_labels_copy = copy.copy(test_labels)
>>> np.random.shuffle(test_labels_copy)
>>> hits_array = np.array(test_labels) == np.array(test_labels_copy)
>>> hits_array.mean()
0.18655387355298308
```

As you can see, a random classifier would score around 19% classification accuracy, so the results of our model seem pretty good in that light.

طبقه‌بندی خطوط خبری

تولید پیش‌بینی‌ها بر روی داده‌های جدید

GENERATING PREDICTIONS ON NEW DATA

Generating predictions for new data

```
predictions = model.predict(x_test)
```

```
# Each entry in predictions is a vector of length 46:
>>> predictions[0].shape
(46,)
```

```
# The coefficients in this vector sum to 1:
>>> np.sum(predictions[0])
1.0
```

```
# The largest entry is the predicted class:
# the class with the highest probability:
>>> np.argmax(predictions[0])
4
```

طبقه‌بندی خطوط خبری

روشی متفاوت برای کار با برچسب‌ها و تابع اتلاف

A Different Way to Handle the Labels and The Loss

We mentioned earlier that another way to encode the labels would be to cast them as an integer tensor, like this:

```
y_train = np.array(train_labels)
y_test = np.array(test_labels)
```

The only thing this approach would change is the choice of the loss function. The loss function `categorical_crossentropy`, expects the labels to follow a categorical encoding. With integer labels, you should use `sparse_categorical_crossentropy`:

```
model.compile(optimizer="rmsprop",
              loss="sparse_categorical_crossentropy",
              metrics=["accuracy"])
```

This new loss function is still mathematically the same as `categorical_crossentropy`; it just has a different interface.

طبقه‌بندی خطوط خبری

اهمیت داشتن تعداد کافی بزرگ لایه‌ی میانی

THE IMPORTANCE OF HAVING SUFFICIENTLY LARGE INTERMEDIATE LAYERS

چون خروجی‌های نهایی ۴۶–بعدی هستند،
باید از لایه‌های میانی با کمتر از ۴۶ واحد مخفی اجتناب کنیم.

اکنون، می‌خواهیم ببینیم:
گلوگاه اطلاعاتی باید چه اتفاقی می‌شود؟
[لایه‌های میانی که بسیار کمتر از ۴۶–بعدی هستند، مثلاً ۴–بعدی]

طبقه‌بندی خطوط خبری

مدلی با گلوگاه اطلاعاتی

Listing 4.22 A model with an information bottleneck

```
model = keras.Sequential([
    layers.Dense(64, activation="relu"),
    layers.Dense(4, activation="relu"),
    layers.Dense(46, activation="softmax")
])
model.compile(optimizer="rmsprop",
              loss="categorical_crossentropy",
              metrics=["accuracy"])
model.fit(partial_x_train,
          partial_y_train,
          epochs=20,
          batch_size=128,
          validation_data=(x_val, y_val))
```

The network now peaks at ~71% validation accuracy, an 8% absolute drop.
This drop is mostly due to the fact that you're trying to compress a lot of information
(enough information to recover the separation hyperplanes of 46 classes)
into an intermediate space that is too low-dimensional.
The network is able to cram **most** of the necessary information
into these eight-dimensional representations, but not all of it.

طبقه‌بندی خطوط خبری

آزمایش‌های بیشتر

FURTHER EXPERIMENTS

The following experiments will help convince you that the architecture choices you've made are all fairly reasonable, although there's still room for improvement:

- Try using larger or smaller layers: 32 units, 128 units, and so on.
- You used two hidden layers.
  Now try using a single hidden layer, or three hidden layers.

طبقه‌بندی خطوط خبری

جمع‌بندی

Here's what you should take away from this example:

- If you're trying to classify data points among *N* classes, your network should end with a Dense layer of size *N*.
- In a single-label, multiclass classification problem, your network should end with a softmax activation so that it will output a probability distribution over the *N* output classes.
- Categorical crossentropy is almost always the loss function you should use for such problems. It minimizes the distance between the probability distributions output by the network and the true distribution of the targets.
- There are two ways to handle **labels** in multiclass classification:
  - Encoding the labels via categorical encoding (also known as one-hot encoding) and using categorical_crossentropy as a loss function
  - Encoding the labels as integers and using the sparse_categorical_crossentropy loss function
- If you need to classify data into a large number of categories, you should avoid creating information bottlenecks in your network due to intermediate layers that are too small.

یادگیری عمیق

مروری بر شبکه‌های عصبی مصنوعی

۶

رگرسیون

مثال: پیش‌بینی قیمت مسکن

پیش‌بینی قیمت مسکن

یک مثال برای رگرسیون

PREDICTING HOUSE PRICES: A REGRESSION EXAMPLE

یک نوع متداول دیگر از مسائل یادگیری ماشینی، **رگرسیون** است؛
که در آن یک مقدار پیوسته به‌جای یک برچسب گسسته پیش‌بینی می‌شود.

مانند:
پیش‌بینی دمای هوای فردا با داشتن داده‌های هواشناسی،
پیش‌بینی زمان تکمیل یک پروژه‌ی نرم‌افزاری با داشتن مشخصه‌های آن.

هدف در این مسئله:
پیش‌بینی قیمت متوسط خانه‌ها در یک محله‌ی داده شده بوستون در اواسط دهه‌ی 1970،
با در اختیار داشتن نقاط داده‌ای در مورد آن محله در آن زمان،
مانند نرخ جرم، نرخ مالیات محلی و ...

NOTE: Don't confuse regression and the algorithm *logistic regression*. Confusingly, logistic regression isn't a regression algorithm—it's a classification algorithm.

پیش‌بینی قیمت مسکن

مجموعه داده‌ی قیمت مسکن بوستون

THE BOSTON HOUSING PRICE DATASET

```
CRIM       0
ZN         0
INDUS      0
CHAS       0
NOX        0
RM         0
AGE        0
DIS        0
RAD        0
TAX        0
PTRATIO    0
B          0
LSTAT      0
MEDV       0
dtype: int64
```

Boston Housing Price Dataset

It has relatively few data points: only 506,
split between 404 training samples and 102 test samples.
And each feature in the input data (for example, the crime rate) has a different scale.
For instance, some values are proportions, which take values between 0 and 1; others take values between 1 and 12, others between 0 and 100, and so on.

```
Boston House Prices dataset

Notes
------
Data Set Characteristics:

    :Number of Instances: 506

    :Number of Attributes: 13 numeric/categorical predictive

    :Median Value (attribute 14) is usually the target

    :Attribute Information (in order):
        - CRIM     per capita crime rate by town
        - ZN       proportion of residential land zoned for lots over 25,000 sq.ft.
        - INDUS    proportion of non-retail business acres per town
        - CHAS     Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
        - NOX      nitric oxides concentration (parts per 10 million)
        - RM       average number of rooms per dwelling
        - AGE      proportion of owner-occupied units built prior to 1940
        - DIS      weighted distances to five Boston employment centres
        - RAD      index of accessibility to radial highways
        - TAX      full-value property-tax rate per $10,000
        - PTRATIO  pupil-teacher ratio by town
        - B        1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
        - LSTAT    % lower status of the population
        - MEDV     Median value of owner-occupied homes in $1000's

    :Missing Attribute Values: None

    :Creator: Harrison, D. and Rubinfeld, D.L.

This is a copy of UCI ML housing dataset.
http://archive.ics.uci.edu/ml/datasets/Housing


This dataset was taken from the StatLib library which is maintained at Carnegie Mellon University.

The Boston house-price data of Harrison, D. and Rubinfeld, D.L. 'Hedonic
```

پیش‌بینی قیمت مسکن

خواندن مجموعه‌داده‌ی Boston Housing Price

Listing 4.23 Loading the Boston housing dataset

```
from tensorflow.keras.datasets import boston_housing
(train_data, train_targets), (test_data, test_targets) = (
    boston_housing.load_data())
```

۴۰۴ نمونه‌ی آموزشی و ۱۰۲ نمونه‌ی آزمایشی داریم،
که هر یک ۱۳ ویژگی عددی دارند (مانند: سرانه نرخ جرم، متوسط تعداد اتاق در خانه، دسترسی به بزرگراه، ...).
تارگت، مقدار میانه‌ی قیمت مسکن اشغال شده توسط مالک بر حسب هزار دلار است.

```
>>> train_data.shape
(404, 13)
>>> test_data.shape
(102, 13)
```

```
>>> train_targets
[ 15.2, 42.3, 50. ... 19.4, 19.4, 29.1]
```

پیش‌بینی قیمت مسکن

آماده‌سازی داده‌ها

PREPARING THE DATA

وارد کردن مقادیری که بازه‌های به‌شدت متفاوتی دارند به شبکه‌ی عصبی می‌تواند مشکل‌ساز باشد.
ممکن است شبکه بتواند به‌صورت خودکار با چنین داده‌های ناهمگنی وفق پیدا کند،
اما این وضعیت قطعاً یادگیری را دشوارتر خواهد کرد.
⇐ باید داده‌ها را نرمال‌سازی ویژگی–به–ویژگی کنیم.

نرمال‌سازی ویژگی–به–ویژگی
Feature-wise Normalization

از هر ویژگی در مجموعه‌داده، میانگین آن را کم و بر انحراف استاندارد آن تقسیم می‌کنیم ⇐ میانگین = صفر، انحراف استاندارد = یک

Listing 4.24 Normalizing the data

```
mean = train_data.mean(axis=0)
train_data -= mean
std = train_data.std(axis=0)
train_data /= std
test_data -= mean
test_data /= std
```

کمیت‌های لازم برای نرمال‌سازی داده‌ها، صرفاً باید با استفاده از داده‌های آموزشی محاسبه شوند.
هیچ کمیتی در کل فرآیند آموزش نباید از داده‌های آزمایشی استفاده کند.

نرمال‌سازی داده‌ها

تکنیک‌ها

DATA NORMALIZATION: TECHNIQUES

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

بازمقیاس‌دهی (نرمال‌سازی *Min-Max*)
Rescaling (Min-Max Normalization)

$$x' = \frac{x - \text{average}(x)}{\max(x) - \min(x)}$$

نرمال‌سازی با میانگین
Mean Normalization

$$x' = \frac{x - \text{average}(x)}{\text{std}(x)}$$

استاندارد سازی
Standardization

پیش‌بینی قیمت مسکن

ساخت شبکه

BUILDING THE NETWORK

از آنجا که تعداد کمی نمونه موجود است،
باید از شبکه‌ی بسیار کوچکی با دو لایه‌ی پنهان استفاده کنیم که هر یک ۶۴ واحد دارند.

در حالت کلی: هر چه تعداد داده‌های آموزشی کمتر باشد، بیش‌برازش بدتری اتفاق می‌افتد.
استفاده از یک شبکه‌ی کوچک راهی برای کاهش بیش‌برازش است.

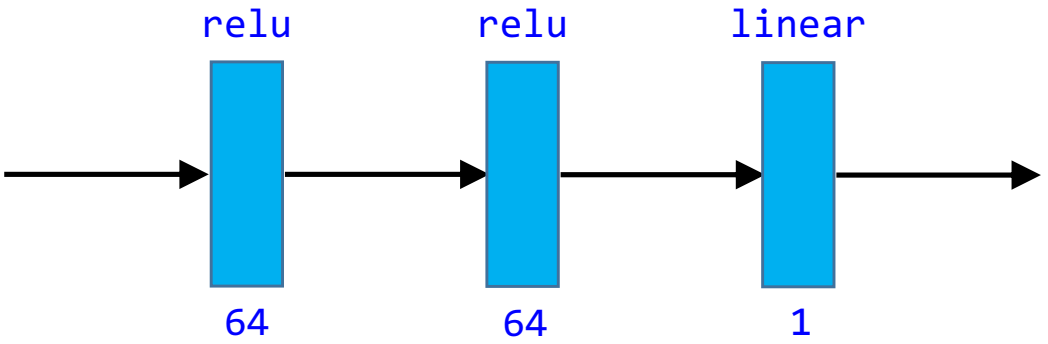
Listing 4.25 Model definition

```
def build_model():
    model = keras.Sequential([
        layers.Dense(64, activation="relu"),
        layers.Dense(64, activation="relu"),
        layers.Dense(1)
    ])
    model.compile(optimizer="rmsprop", loss="mse", metrics=["mae"])
    return model
```

Because we need to instantiate the same model multiple times, we use a function to construct it.

پیش‌بینی قیمت مسکن

ساخت شبکه



این شبکه با یک واحد تنها بدون تابع فعالیت (یعنی تابع فعالیت خطی f(x) = x) خاتمه یافته است.

این یک چیدمان معمول برای رگرسیون اسکالر است.

اگر در لایه‌ی آخر تابع فعالیتی مانند sigmoid قرار می‌گرفت،
شبکه تنها می‌توانست پیش‌بینی مقادیر بین 0 و 1 را یاد بگیرد.
اما در اینجا چون لایه‌ی آخر کاملاً خطی است،
شبکه آزاد است که پیش‌بینی مقادیر در هر بازه‌ای را یاد بگیرد.

پیش‌بینی قیمت مسکن

تابع اتلاف MSE و متریک نظارت MAE

تابع اتلاف مناسب برای کاربردهای رگرسیون، MSE است.
برای نظارت در حین آموزش، متریک مناسب MAE است.

خطای میانگین مربعات
Mean Squared Error (MSE)

میانگین مجذور
مقادیر پیش‌بینی شده و تارگت‌ها

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

خطای میانگین قدر مطلق‌ها
Mean Absolute Error (MAE)

میانگین قدر مطلق
مقادیر پیش‌بینی شده و تارگت‌ها

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$$

Mean squared error	$\mathrm{MSE} = \frac{1}{n}\sum_{t=1}^{n} e_t^2$
Root mean squared error	$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{t=1}^{n} e_t^2}$
Mean absolute error	$\mathrm{MAE} = \frac{1}{n}\sum_{t=1}^{n} \lvert e_t \rvert$
Mean absolute percentage error	$\mathrm{MAPE} = \frac{100\%}{n}\sum_{t=1}^{n} \left\lvert \frac{e_t}{y_t} \right\rvert$

پیش‌بینی قیمت مسکن

اعتبارسنجی روی‌کرد با استفاده از تکنیک اعتبارسنجی *K*-fold

VALIDATING THE APPROACH USING K-FOLD VALIDATION

برای نظارت بر دقت مدل «بر روی داده‌هایی که هرگز تاکنون دیده نشده‌اند» در حین آموزش،
باید یک **مجموعه‌ی اعتبارسنجی** ایجاد کنیم.
اما با توجه به اینکه تعداد داده‌های این مثال اندک است، مجموعه‌ی اعتبارسنجی بسیار کوچک خواهد شد
و این باعث می‌شود امتیازات اعتبارسنجی به داده‌های انتخاب شده برای اعتبارسنجی بسیار وابسته شود،
یعنی: امتیازات اعتبارسنجی نسبت به جداسازی‌ها دارای واریانس بالایی خواهند شد.
و این باعث می‌شود ارزیابی مدل دارای قابلیت اطمینان بالایی نباشد.

در چنین موقعیت‌هایی، استفاده از تکنیک *K-fold* cross-validation متداول است.

پیش‌بینی قیمت مسکن

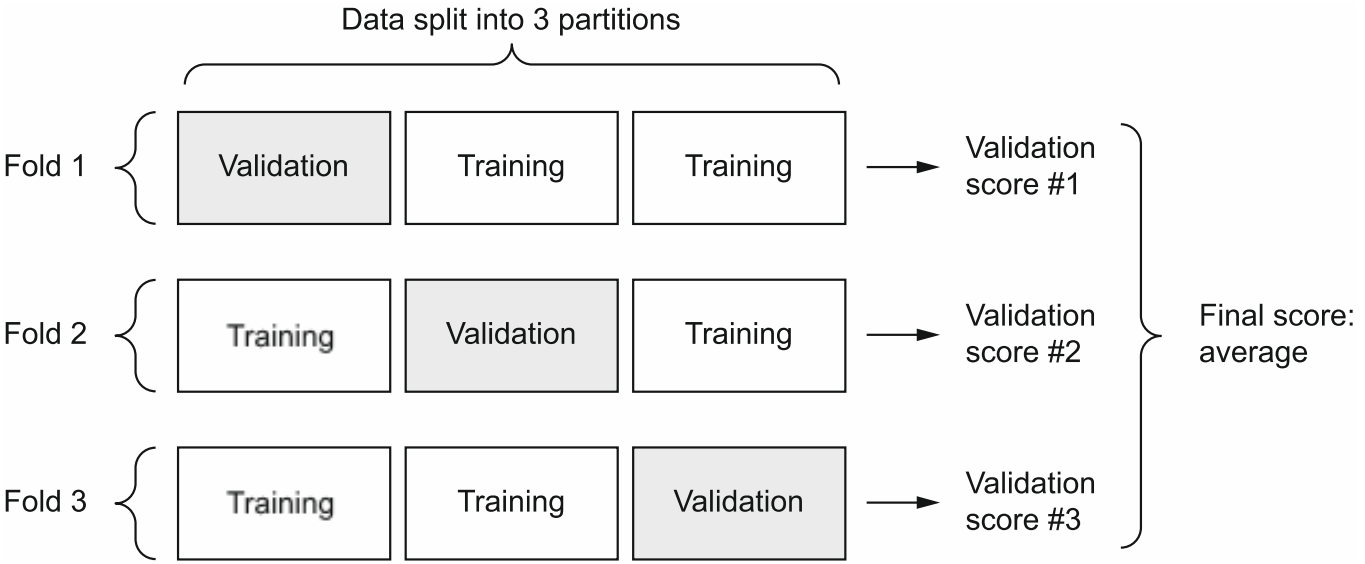
اعتبار سنجی با تکنیک K-fold Cross-Validation

K-FOLD CROSS-VALIDATION

تکنیک *K-fold* cross-validation:

داده‌های موجود را به K سلول افراز می‌کنیم (معمولاً $K = 4,5$)،
K مدل یکسان را نمونه‌سازی می‌کنیم
و هر یک را بر روی $K - 1$ سلول آموزش می‌دهیم و بر روی سلول باقیمانده ارزیابی می‌کنیم.

امتیاز اعتبارسنجی مدل = متوسط K امتیاز اعتبارسنجی حاصل.



3-fold cross-validation

پیش‌بینی قیمت مسکن

اعتبار سنجی با تکنیک K-fold Cross-Validation

Listing 4.26 K-fold validation

```
k = 4
num_val_samples = len(train_data) // k
num_epochs = 100
all_scores = []
for i in range(k):
    print(f"Processing fold #{i}")
    val_data = train_data[i * num_val_samples: (i + 1) * num_val_samples]
    val_targets = train_targets[i * num_val_samples: (i + 1) * num_val_samples]
    partial_train_data = np.concatenate(
        [train_data[:i * num_val_samples],
         train_data[(i + 1) * num_val_samples:]],
        axis=0)
    partial_train_targets = np.concatenate(
        [train_targets[:i * num_val_samples],
         train_targets[(i + 1) * num_val_samples:]],
        axis=0)
    model = build_model()
    model.fit(partial_train_data, partial_train_targets,
              epochs=num_epochs, batch_size=16, verbose=0)
    val_mse, val_mae = model.evaluate(val_data, val_targets, verbose=0)
    all_scores.append(val_mae)
```

Prepares the validation data: data from partition #k

Prepares the training data: data from all other partitions

Builds the Keras model (already compiled)

Trains the model (in silent mode, verbose = 0)

Evaluates the model on the validation data

پیش‌بینی قیمت مسکن

اعتبار سنجی با تکنیک K-fold Cross-Validation

```
# Running this with num_epochs = 100 yields the following results:

>>> all_scores
[2.112449, 3.0801501, 2.6483836, 2.4275346]

>>> np.mean(all_scores)
2.5671294
```

The different runs do indeed show rather different validation scores, from 2.1 to 3.1. The average (2.6) is a much more reliable metric than any single score—that's the entire point of K-fold cross-validation. In this case, you're off by $2,600 on average, which is significant considering that the prices range from $10,000 to $50,000.

پیش‌بینی قیمت مسکن

آموزش طولانی‌تر شبکه (۵۰۰ اپک) + ذخیره‌سازی امتیاز اعتبارسنجی در پایان هر اپک

Listing 4.27 Saving the validation logs at each fold

```
num_epochs = 500
all_mae_histories = []
for i in range(k):
    print(f"Processing fold #{i}")
    val_data = train_data[i * num_val_samples: (i + 1) * num_val_samples]
    val_targets = train_targets[i * num_val_samples: (i + 1) * num_val_samples]
    partial_train_data = np.concatenate(
        [train_data[:i * num_val_samples],
         train_data[(i + 1) * num_val_samples:]],
        axis=0)
    partial_train_targets = np.concatenate(
        [train_targets[:i * num_val_samples],
         train_targets[(i + 1) * num_val_samples:]],
        axis=0)
    model = build_model()
    history = model.fit(partial_train_data, partial_train_targets,
                        validation_data=(val_data, val_targets),
                        epochs=num_epochs, batch_size=16, verbose=0)
    mae_history = history.history["val_mae"]
    all_mae_histories.append(mae_history)
```

Prepares the validation data: data from partition #k

Prepares the training data: data from all other partitions

Builds the Keras model (already compiled)

Trains the model (in silent mode, verbose=0)

پیش‌بینی قیمت مسکن

آموزش طولانی‌تر شبکه (۵۰۰ اپک) + ذخیره‌سازی امتیاز اعتبارسنجی در پایان هر اپک

You can then compute the average of the per-epoch MAE scores for all folds.

Listing 4.28 Building the history of successive mean K-fold validation scores

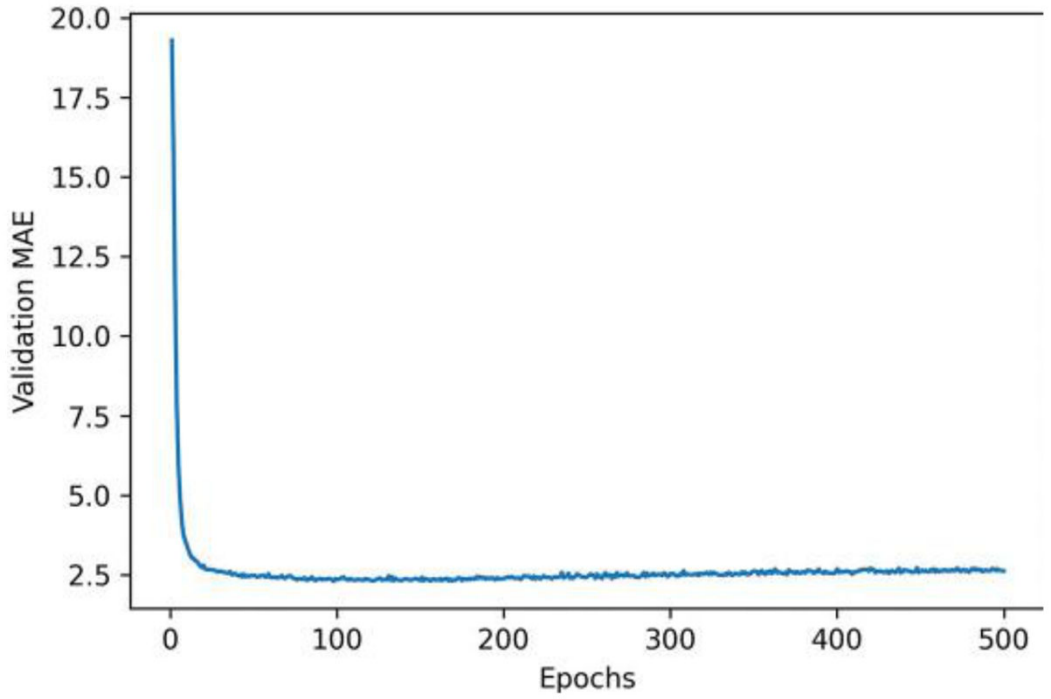
```
average_mae_history = [
    np.mean([x[i] for x in all_mae_histories]) for i in range(num_epochs)]
```

پیش‌بینی قیمت مسکن

رسم امتیازات اعتبارسنجی

Listing 4.29 Plotting validation scores

```
plt.plot(range(1, len(average_mae_history) + 1), average_mae_history)
plt.xlabel("Epochs")
plt.ylabel("Validation MAE")
plt.show()
```



پیش‌بینی قیمت مسکن

رسم امتیازات اعتبارسنجی [در بازه‌ی متفاوت برای مشاهده‌ی بهتر]

Listing 4.30 Plotting validation scores, excluding the first 10 data points

```
truncated_mae_history = average_mae_history[10:]
plt.plot(range(1, len(truncated_mae_history) + 1), truncated_mae_history)
plt.xlabel("Epochs")
plt.ylabel("Validation MAE")
plt.show()
```

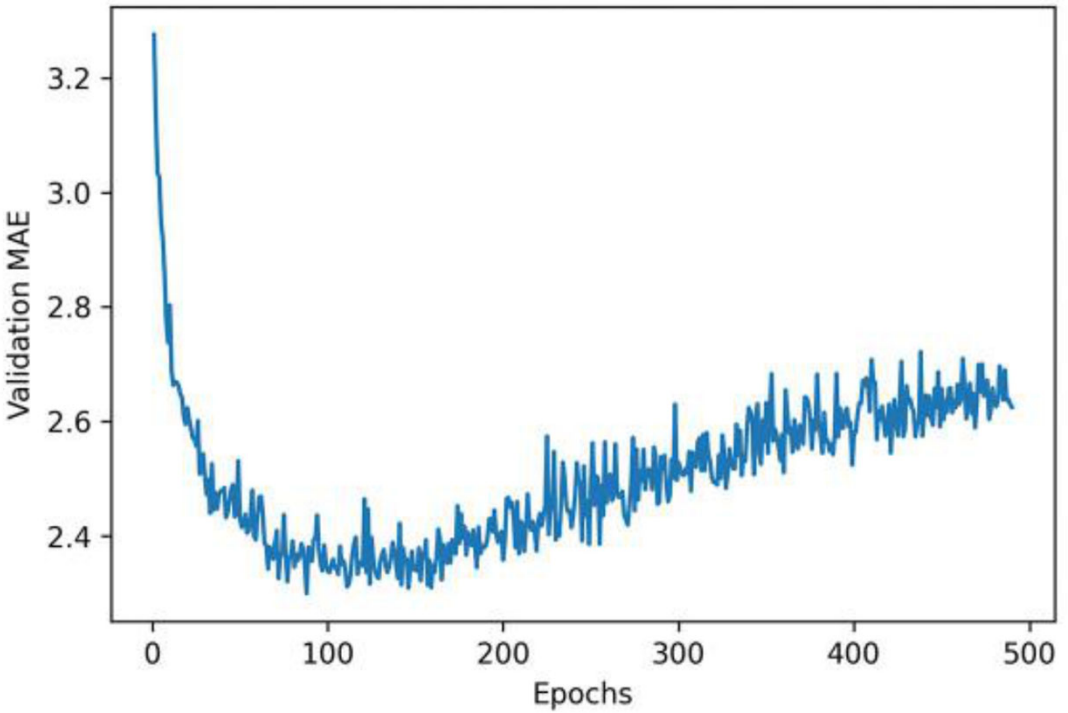
- Omit the first 10 data points, which are on a different scale than the rest of the curve.
- Replace each point with an exponential moving average of the previous points, to obtain a smooth curve.

exponential moving average

$$EMA = Price(t) * k + EMA(y) * (1 - k)$$

پیش‌بینی قیمت مسکن

رسم امتیازات اعتبارسنجی [در بازه‌ی متفاوت برای مشاهده‌ی بهتر]



Validation **MAE** by epoch, excluding the first 10 data points

بر اساس این نمودار، بهبود چشمگیر امتیازات MAE اعتبارسنجی پس از ۱۳۰ تا ۱۴۰ اپک متوقف می‌شود.
پس از عبور از این نقطه، **بیش‌برازش** شروع می‌شود.

پیش‌بینی قیمت مسکن

آموزش مدل نهایی

Listing 4.31 Training the final model

Gets a fresh, compiled model

Trains it on the entirety of the data

```
model = build_model()
model.fit(train_data, train_targets,
          epochs=130, batch_size=16, verbose=0)
test_mse_score, test_mae_score = model.evaluate(test_data, test_targets)
```

```
# Here's the final result:

>>> test_mae_score
2.4642276763916016
```

We're still off by about $2,500.

4.3.5 Generating predictions on new data

When calling `predict()` on our binary classification model, we retrieved a scalar score between 0 and 1 for each input sample. With our multiclass classification model, we retrieved a probability distribution over all classes for each sample. Now, with this scalar regression model, `predict()` returns the model's guess for the sample's price in thousands of dollars:

```
>>> predictions = model.predict(test_data)
>>> predictions[0]
array([9.990133], dtype=float32)
```

The first house in the test set is predicted to have a price of about $10,000.

پیش‌بینی قیمت مسکن

جمع‌بندی

Here's what you should take away from this example:

- Regression is done using different loss functions than what we used for classification.
  - Mean squared error (MSE) is a loss function commonly used for regression.
  - Similarly, evaluation metrics to be used for regression differ from those used for classification; naturally, the concept of accuracy doesn't apply for regression. A common regression metric is mean absolute error (MAE).
- When features in the input data have values in different ranges, each feature should be scaled independently as a preprocessing step.
- When there is *little data available*, using K-fold validation is a great way to reliably evaluate a model.
- When little training data is available, it's preferable to use a small network with few hidden layers (typically only one or two), in order to avoid severe overfitting.

خلاصه

- You'll usually need to preprocess raw data before feeding it into a neural network.
- When your data has features with different ranges, scale each feature independently as part of preprocessing.
- As training progresses, neural networks eventually begin to overfit and obtain worse results on never-before-seen data.
- If you don't have much training data, use a small network with only one or two hidden layers, to avoid severe overfitting.
- If your data is divided into many categories, you may cause information bottlenecks if you make the intermediate layers too small.
- Regression uses different loss functions and different evaluation metrics than classification.
- When you're working with little data, K-fold validation can help reliably evaluate your model.

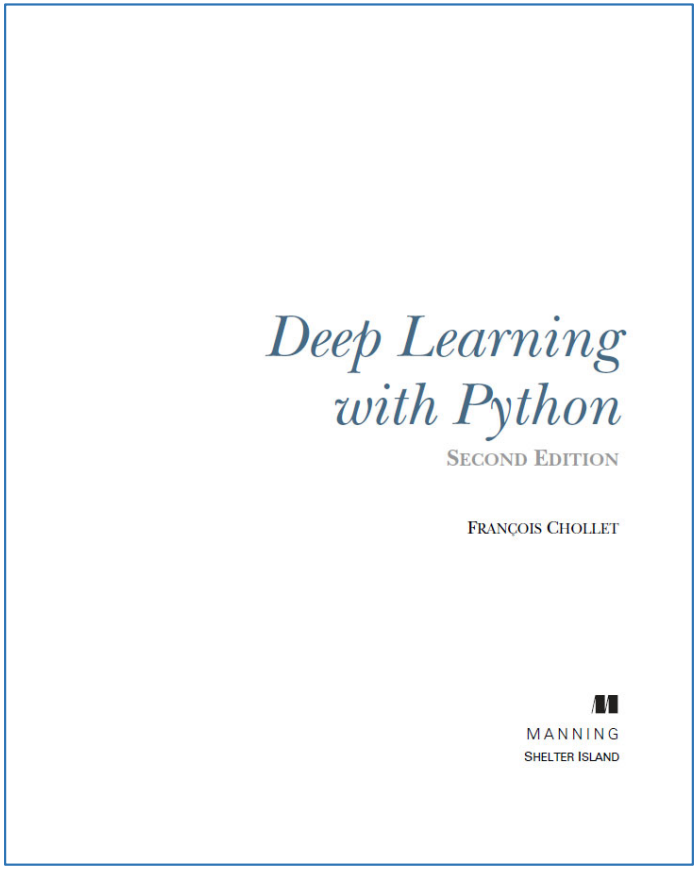
یادگیری عمیق

مروری بر شبکه‌های عصبی مصنوعی

۷

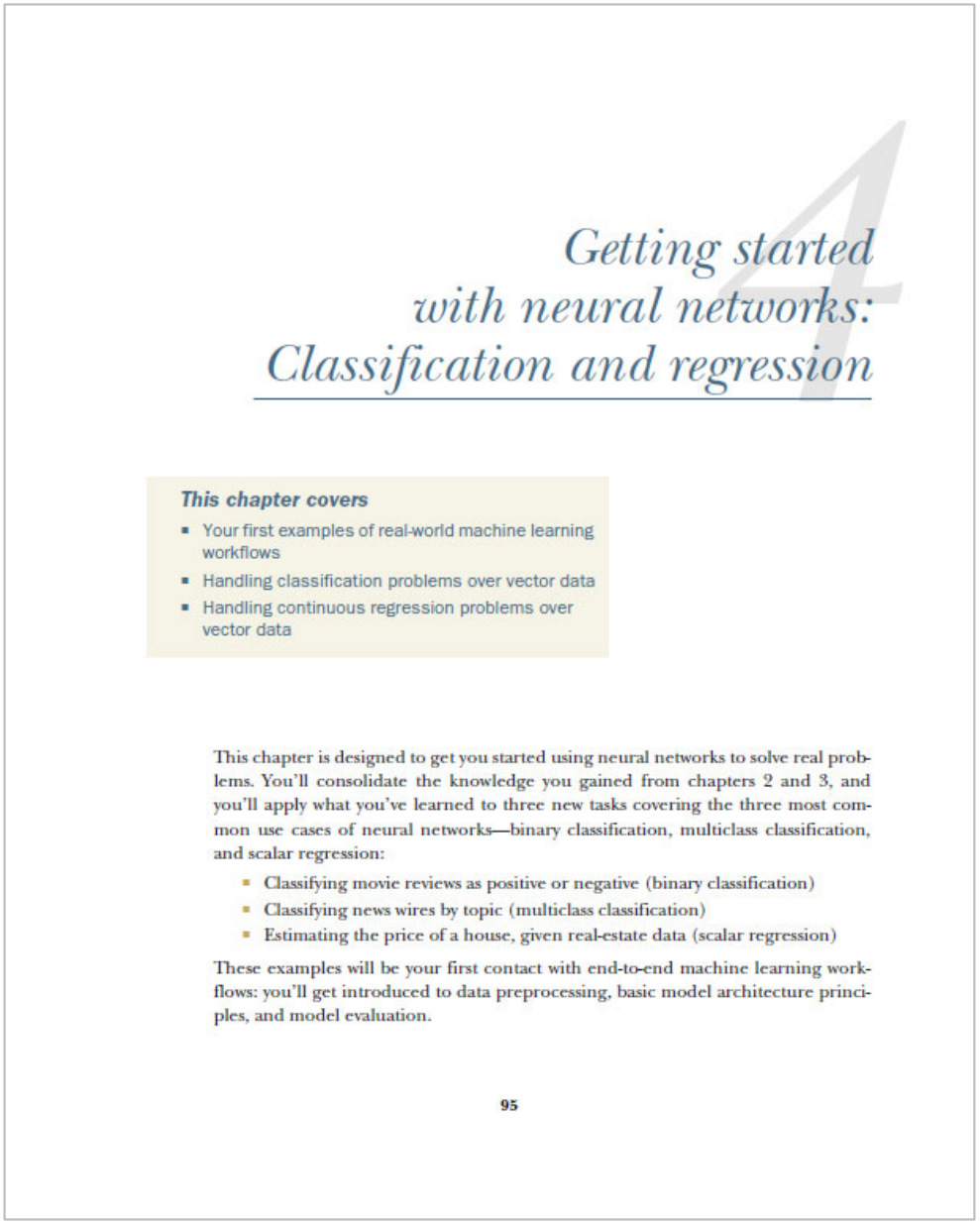
منابع

منبع اصلی



François Chollet,
Deep Learning with Python,
Second Edition, Manning Publications, 2021.

Chapter 4



4 Getting started with neural networks: Classification and regression

This chapter covers

- Your first examples of real-world machine learning workflows
- Handling classification problems over vector data
- Handling continuous regression problems over vector data

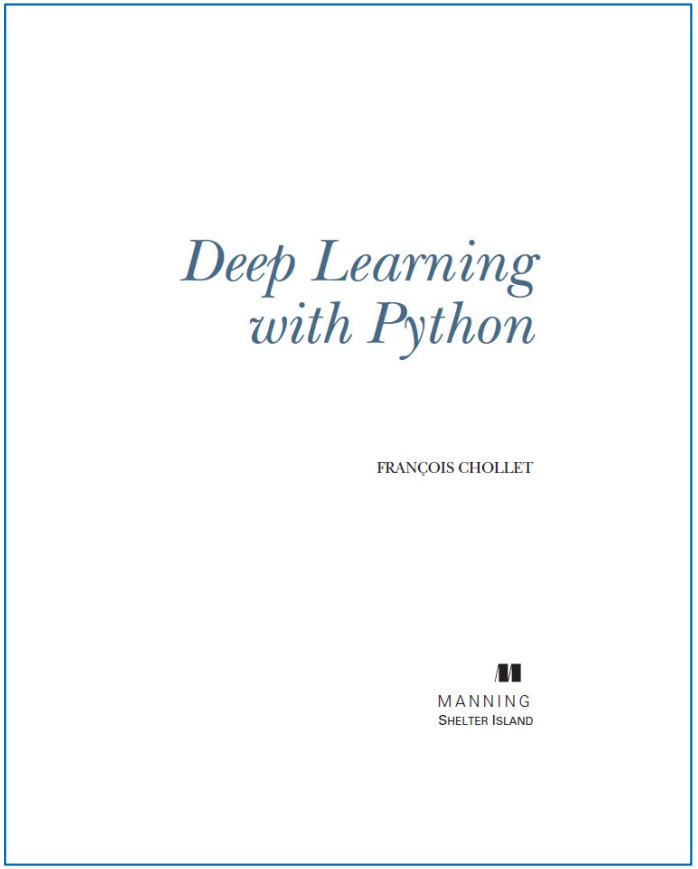
This chapter is designed to get you started using neural networks to solve real problems. You'll consolidate the knowledge you gained from chapters 2 and 3, and you'll apply what you've learned to three new tasks covering the three most common use cases of neural networks—binary classification, multiclass classification, and scalar regression:

- Classifying movie reviews as positive or negative (binary classification)
- Classifying news wires by topic (multiclass classification)
- Estimating the price of a house, given real-estate data (scalar regression)

These examples will be your first contact with end-to-end machine learning workflows: you'll get introduced to data preprocessing, basic model architecture principles, and model evaluation.

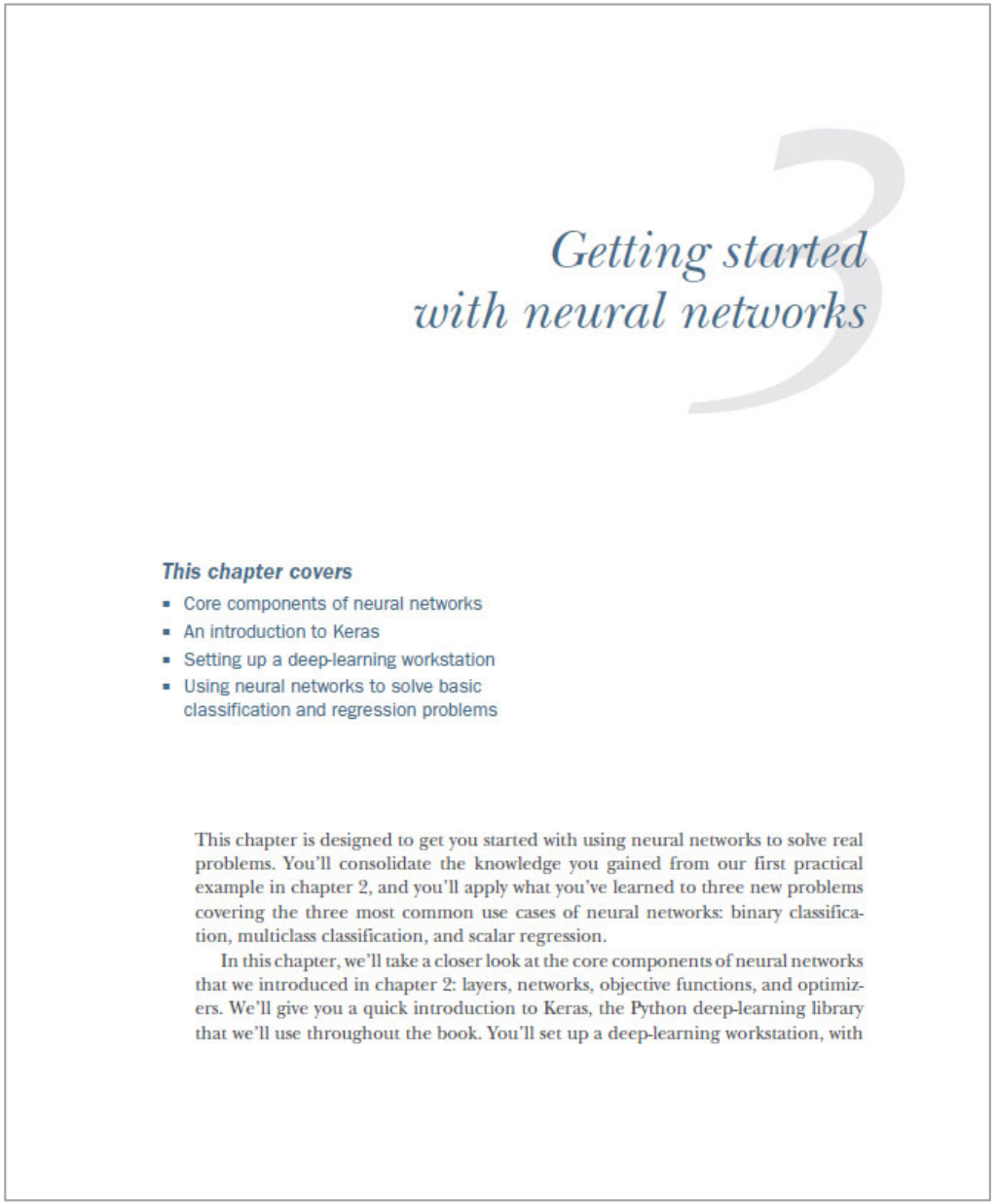
95

منبع اصلی



François Chollet,
Deep Learning with Python,
Manning Publications, 2018.

Chapter 3



3 Getting started with neural networks

This chapter covers

- Core components of neural networks
- An introduction to Keras
- Setting up a deep-learning workstation
- Using neural networks to solve basic classification and regression problems

This chapter is designed to get you started with using neural networks to solve real problems. You'll consolidate the knowledge you gained from our first practical example in chapter 2, and you'll apply what you've learned to three new problems covering the three most common use cases of neural networks: binary classification, multiclass classification, and scalar regression.

In this chapter, we'll take a closer look at the core components of neural networks that we introduced in chapter 2: layers, networks, objective functions, and optimizers. We'll give you a quick introduction to Keras, the Python deep-learning library that we'll use throughout the book. You'll set up a deep-learning workstation, with