یادگیری عمیق

جلسه ۱۲

مبانی یادگیری ماشینی (۲)

Fundamentals of Machine Learning (2)

کاظم فولادی قلعه

دانشکده مهندسی، دانشکدگان فارابی

دانشگاه تهران

http://courses.fouladi.ir/deep

یادگیری عمیق

مبانی یادگیری ماشینی

۱

بیش‌برازش و کم‌برازش

بیش‌برازش و کم‌برازش

OVERFITTING AND UNDERFITTING

بیش‌برازش در هر مسئله‌ی یادگیری ماشینی اتفاق می‌افتد.
درک چگونگی رویارویی با بیش‌بردازش برای تسلط بر یادگیری ماشینی ضروری است.

مسئله‌ی بنیادین در یادگیری ماشینی، کشمکش میان **بهینه‌سازی (سازگاری)** و **تعمیم** است.

هدف این بازی رسیدن به تعمیم خوب است،
اما ما تعمیم را کنترل نمی‌کنیم، بلکه ما فقط می‌توانیم مدل را بر اساس داده‌های آموزشی تنظیم کنیم.

در ابتدای آموزش، بهینه‌سازی و تعمیم با یکدیگر همبسته هستند:
هر چه اتلاف روی داده‌های آموزشی کمتر باشد، روی داده‌های آزمایشی هم کمتر است (وضعیت **کم‌برازش**).
اما پس از چند تکرار مشخص روی داده‌های آموزشی، بهبود تعمیم متوقف می‌شود (شروع **بیش‌برازش**).
مدل شروع می‌کند به یادگیری الگوهایی که مختص داده‌های آموزشی است (آنها را حفظ می‌کند)؛
اما هنگامی که داده‌های جدید مطرح می‌شوند، این الگوها گمراه‌کننده یا نامربوط هستند.

سازگاری و تعمیم

سازگاری *Consistency*	تعمیم *Generalization*
یک فرضیه سازگار است اگر بر روی همه‌ی نمونه‌های آموزشی درست باشد.	یک فرضیه تعمیم‌پذیر است اگر مقادیر خروجی نمونه‌های جدید را به‌درستی پیش‌بینی کند.
بهینه‌سازی به‌دنبال تنظیم یک مدل برای دستیابی به بهترین کارآیی ممکن بر روی داده‌های آموزشی است.	تعمیم مشخص می‌کند که مدل آموزش‌دیده چه‌قدر خوب بر روی داده‌هایی که پیش از این دیده نشده‌اند عمل می‌کند.

بده‌بستان میان **سازگاری ـ تعمیم‌پذیری**:
فرضیه‌های پیچیده با سازگاری کامل و فرضیه‌های ساده‌تر با تعمیم‌پذیری بالاتر

بیش‌برازش و کم‌برازش

OVERFITTING AND UNDERFITTING

بیش‌برازش *Overfitting*	دقت مدل روی داده‌های آموزشی بیشتر از دقت مدل روی داده‌های اعتبارسنجی شده است.
کم‌برازش *Underfitting*	شبکه هنوز همه‌ی الگوهای مربوط در داده‌های آموزشی را مدل نکرده است.

جلوگیری از بیش‌برازش

رگولاریزاسیون

بهترین راه‌حل برای جلوگیری از بیش‌برازش، استفاده از داده‌های بیشتر می‌باشد.
مدلی که بر روی تعداد داده‌ی بیشتری آموزش دیده باشد، طبیعتاً تعمیم‌پذیری بالاتری خواهد داشت.

بهترین راه‌حل بعدی، تعدیل کمیت اطلاعاتی است که مدل مجاز به ذخیره کردن آنهاست،
یا اضافه کردن قیدها بر روی اطلاعاتی است که مدل مجاز به ذخیره کردن آنهاست.

اگر یک شبکه فقط برای حفظ کردن تعداد کوچکی از الگوها توانایی داشته باشد،
فرآیند بهینه‌سازی آن را مجبور خواهد کرد که بر روی برجسته‌ترین الگوها تمرکز کند،
که این شانس بهتری برای تعمیم‌دهی بالاتر دارد.

رگولاریزاسیون *Regularization*	فرآیند مقابله با بیش‌برازش از طریق تعدیل / منظم‌سازی وزن‌ها

یادگیری عمیق

مبانی یادگیری ماشینی

۲

بهبود تعمیم‌پذیری

بهبود تعمیم‌پذیری

با استفاده از داده‌های آموزشی بیشتر / بهتر

استفاده از داده‌های آموزشی بیشتر / بهتر

Get more / better training data

5.4 Improving generalization

Once your model has shown itself to have some generalization power and to be able to overfit, it's time to switch your focus to maximizing generalization.

5.4.1 Dataset curation

You've already learned that generalization in deep learning originates from the latent structure of your data. If your data makes it possible to smoothly interpolate between samples, you will be able to train a deep learning model that generalizes. If your problem is overly noisy or fundamentally discrete, like, say, list sorting, deep learning will not help you. Deep learning is curve fitting, not magic.

As such, it is essential that you make sure that you're working with an appropriate dataset. Spending more effort and money on data collection almost always yields a much greater return on investment than spending the same on developing a better model.

- Make sure you have enough data. Remember that you need a *dense sampling* of the input-cross-output space. More data will yield a better model. Sometimes, problems that seem impossible at first become solvable with a larger dataset.
- Minimize labeling errors—visualize your inputs to check for anomalies, and proofread your labels.
- Clean your data and deal with missing values (we'll cover this in the next chapter).
- If you have many features and you aren't sure which ones are actually useful, do feature selection.

A particularly important way to improve the generalization potential of your data is *feature engineering*. For most machine learning problems, feature engineering is a key ingredient for success. Let's take a look.

بهبود تعمیم‌پذیری

با توسعه‌ی ویژگی‌های بهتر

توسعه‌ی ویژگی‌های بهتر

Develop better features

5.4.2 Feature engineering

Feature engineering is the process of using your own knowledge about the data and about the machine learning algorithm at hand (in this case, a neural network) to make the algorithm work better by applying hardcoded (non-learned) transformations to the data before it goes into the model. In many cases, it isn't reasonable to expect a machine learning model to be able to learn from completely arbitrary data. The data needs to be presented to the model in a way that will make the model's job easier.

Let's look at an intuitive example. Suppose you're trying to develop a model that can take as input an image of a clock and can output the time of day (see figure 5.16).

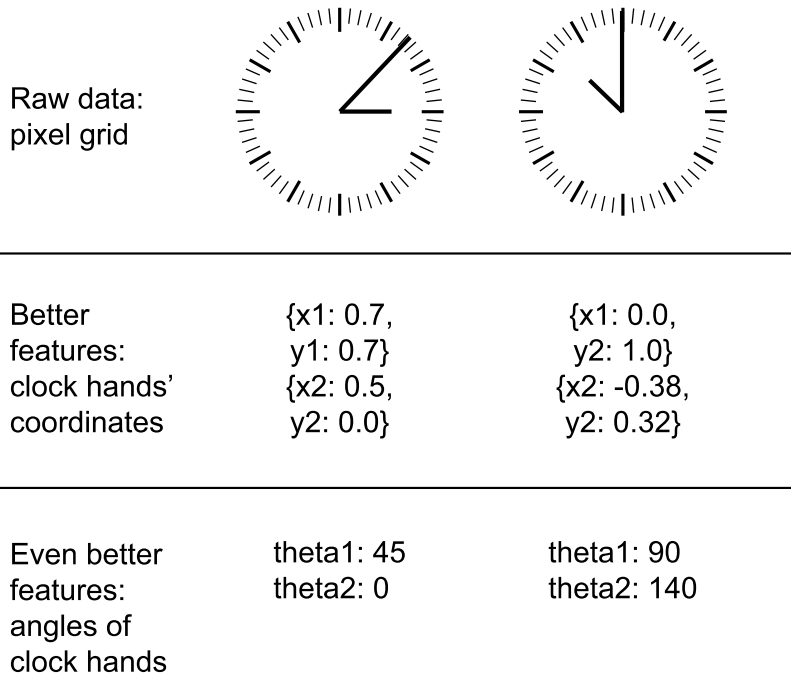


Figure 5.16 Feature engineering for reading the time on a clock

بهبود تعمیم‌پذیری

با مهندسی ویژگی‌ها

FEATURE ENGINEERING

استخراج ویژگی
Feature Extraction

در مهندسی ویژگی‌ها،
از دانایی موجود در مورد داده‌ها و الگوریتم یادگیری ماشینی مورد نظر (در اینجا شبکه‌های عصبی) استفاده می‌شود و تبدیل‌هایی بر روی داده‌ها پیش از ورود به مدل از طریق کدنویسی اعمال می‌شود تا موجب بهبود عملکرد الگوریتم شود.

در بسیاری از موارد، منطقی نیست که توقع داشته باشیم یک مدل یادگیری ماشینی قادر باشد از **داده‌های کاملاً دلخواه** یاد بگیرد.

داده‌ها باید به‌شیوه‌ای به مدل ارائه شوند که کار مدل را ساده‌تر کنند.

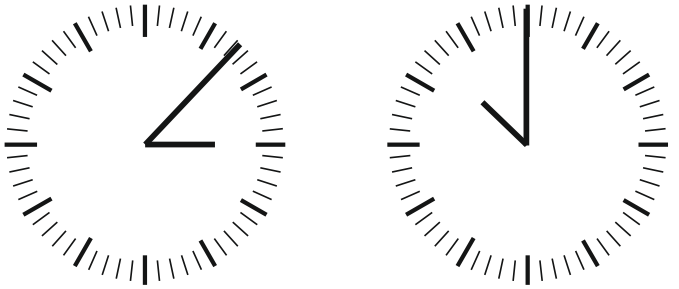
بهبود تعمیم‌پذیری

با مهندسی ویژگی: مثال (خواندن زمان از روی یک ساعت)

فرض می‌کنیم می‌خواهیم مدلی توسعه بدهیم که
تصویر یک ساعت را به‌عنوان **ورودی** دریافت کند و زمان روز را به‌عنوان **خروجی** تحویل بدهد.

اگر پیکسل‌های خام تصویر را به‌عنوان ورودی در نظر بگیریم، یا یک مسئله‌ی دشوار یادگیری ماشینی سروکار خواهیم داشت.
(نیاز به CNN با منابع محاسباتی بالا و داده‌های آموزشی بسیار زیاد)

Raw data: pixel grid



اگر این مسئله را در سطح بالا درک کنیم (روش خواندن زمان از روی ساعت توسط انسان‌ها) می‌توانیم ویژگی‌های ورودی بهتری برای الگوریتم یادگیری بیابیم (مثل: مختصات نوک عقربه‌ها)

Better features: clock hands' coordinates	{x1: 0.7, y1: 0.7} {x2: 0.5, y2: 0.0}	{x1: 0.0, y2: 1.0} {x2: -0.38, 2: 0.32}

حتی می‌توانیم از مختصات قطبی استفاده کنیم که در این‌صورت ویژگی‌های لازم، زاویه‌ی هر عقربه است. در اینجا که به‌حدی ساده می‌شود که دیگر حتی نیازی به یادگیری ماشینی نیست!

Even better features: angles of clock hands	theta1: 45 theta2: 0	theta1: 90 theta2: 140

بهبود تعمیم‌پذیری

با مهندسی ویژگی: ضرورت

اساس مهندسی ویژگی‌ها: ساده‌ترسازی یک مسئله از طریق بیان آن به‌صورتی ساده‌تر

پیش از یادگیری عمیق، مهندسی ویژگی‌ها بسیار حیاتی بود،
زیرا مدل‌های کم‌عمق فضاهای فرضیه‌ی به‌اندازه‌ی کافی غنی ندارند تا ویژگی‌های مفید را خودشان یادبگیرند؛
خوشبختانه، یادگیری عمیق مدرن نیاز به مهندسی را تا حد بسیاری حذف کرده است:
زیرا این مدل‌ها قادر به استخراج خودکار ویژگی‌های سودمند از داده‌های خام هستند.

اما این به‌آن معنا نیست که در استفاده از شبکه‌های عصبی عمیق به مهندسی ویژگی‌ها نیازی نداریم.

دلایل نیاز به مهندسی ویژگی‌ها در یادگیری عمیق

- **امکان حل مسائل به‌صورتی بهتر با استفاده از منابع کمتر با ویژگی‌های خوب**
- **امکان حل مسائل با حجم داده‌ی بسیار کمتر**

5.4.3 Using early stopping

In deep learning, we always use models that are vastly overparameterized: they have way more degrees of freedom than the minimum necessary to fit to the latent manifold of the data. This overparameterization is not an issue, because *you never fully fit a deep learning model*. Such a fit wouldn't generalize at all. You will always interrupt training long before you've reached the minimum possible training loss.

Finding the exact point during training where you've reached the most generalizable fit—the exact boundary between an underfit curve and an overfit curve—is one of the most effective things you can do to improve generalization.

In the examples in the previous chapter, we would start by training our models for longer than needed to figure out the number of epochs that yielded the best validation metrics, and then we would retrain a new model for exactly that number of epochs. This is pretty standard, but it requires you to do redundant work, which can sometimes be expensive. Naturally, you could just save your model at the end of each epoch, and once you've found the best epoch, reuse the closest saved model you have. In Keras, it's typical to do this with an `EarlyStopping` callback, which will interrupt training as soon as validation metrics have stopped improving, while remembering the best known model state. You'll learn to use callbacks in chapter 7.

5.4.4 Regularizing your model

Regularization techniques are a set of best practices that actively impede the model's ability to fit perfectly to the training data, with the goal of making the model perform better during validation. This is called "regularizing" the model, because it tends to make the model simpler, more "regular," its curve smoother, more "generic"; thus it is less specific to the training set and better able to generalize by more closely approximating the latent manifold of the data.

Keep in mind that regularizing a model is a process that should always be guided by an accurate evaluation procedure. You will only achieve generalization if you can measure it.

Let's review some of the most common regularization techniques and apply them in practice to improve the movie-classification model from chapter 4.

تعمیم‌پذیر کردن شبکه‌های عصبی

برای ایجاد تعمیم‌پذیری، روی‌کرد کلی سعی در یافتن ساده‌ترین شبکه است که بر داده‌ها fit شود.

تکنیک‌های رگولاریزاسیون

(منظم‌سازی)

REGULARIZATION TECHNIQUES

- **کاهش اندازه‌ی شبکه**
  Reducing the network's size
- **افزودن رگولاریزاسیون وزنی**
  Adding weight regularization
- **افزودن برون‌اندازی Drop-out**
  Adding dropout

تکنیک‌های رگولاریزاسیون

کاهش اندازه‌ی شبکه

REDUCING THE NETWORK'S SIZE

ساده‌ترین راه برای اجتناب از بیش‌برازش، کاهش اندازه‌ی مدل است:
تعداد پارامترهای قابل یادگیری در مدل (تابعی از تعداد لایه‌ها و تعداد واحدها در هر لایه)

ظرفیت مدل *Model's Capacity*	تعداد پارامترهای قابل یادگیری در مدل

مدلی که پارامترهای بیشتری دارد، **ظرفیت حفظ‌کردن** بیشتری دارد
و بنابراین، به‌سادگی می‌تواند یک نگاشت دیکشنری مانند بین نمونه‌های آموزشی و تارگت آنها را یاد بگیرد،
که یک نگاشت بدون هرگونه قدرت تعمیم است.
(به‌ویژه در مدل‌های عمیق که ظرفیت آنها بالاست)

از طرف دیگر، اگر شبکه منابع حفظ کردن محدودی داشته باشد،
قادر نخواهد بود این نگاشت را به‌سادگی یاد بگیرد؛
پس برای می‌نیمم‌سازی loss به سمت یادگیری بازنمایی‌های فشرده‌ای کشیده می‌شود که
قدرت پیش‌بینی تارگت‌ها را داشته باشند.

تکنیک‌های رگولاریزاسیون

کاهش اندازه‌ی شبکه: تعادل بین ظرفیت بسیار زیاد و ظرفیت ناکافی

مدل مورد استفاده باید تعداد کافی پارامتر داشته باشد تا دچار **کم‌برازش** نشود.

یک وضعیت بده‌بستان (مصالحه) وجود دارد

که باید بین ظرفیت بسیار زیاد و ظرفیت ناکافی پیدا شود.

هیچ فرمول جادویی برای تعیین تعداد درست لایه‌ها یا اندازه‌ی درست هر لایه وجود ندارد.

برای یافتن اندازه‌ی درست مدل، باید آرایه‌ای از معماری‌های مختلف روی مجموعه‌ی اعتبارسنجی ارزیابی شود.

به‌طور کلی، برای یافتن اندازه‌ی مناسب مدل،
از تعداد نسبتاً کمی لایه و پارامتر شروع می‌کنیم
و اندازه‌ی لایه‌ها را اضافه می‌کنیم یا لایه‌های جدیدی اضافه می‌کنیم
تا با توجه به اتلاف روی داده‌های اعتبارسنجی، افت بازدهی را مشاهده کنیم.

تکنیک‌های رگولاریزاسیون

کاهش اندازه‌ی شبکه: مثال (طبقه‌بندی نقد فیلم‌ها)

Listing 5.10 Original model

```
from tensorflow.keras.datasets import imdb
(train_data, train_labels), _ = imdb.load_data(num_words=10000)

def vectorize_sequences(sequences, dimension=10000):
    results = np.zeros((len(sequences), dimension))
    for i, sequence in enumerate(sequences):
        results[i, sequence] = 1.
    return results
train_data = vectorize_sequences(train_data)
model = keras.Sequential([
    layers.Dense(16, activation="relu"),
    layers.Dense(16, activation="relu"),
    layers.Dense(1, activation="sigmoid")
])
model.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
history_original = model.fit(train_data, train_labels,
                             epochs=20, batch_size=512, validation_split=0.4)
```

تکنیک‌های رگولاریزاسیون

کاهش اندازه‌ی شبکه: مثال (طبقه‌بندی نقد فیلم‌ها)

Now let's try to replace it with this smaller model.

Listing 5.11 Version of the model with lower capacity

```
model = keras.Sequential([
    layers.Dense(4, activation="relu"),
    layers.Dense(4, activation="relu"),
    layers.Dense(1, activation="sigmoid")
])
model.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
history_smaller_model = model.fit(
    train_data, train_labels,
    epochs=20, batch_size=512, validation_split=0.4)
```

تکنیک‌های رگولاریزاسیون

کاهش اندازه‌ی شبکه : مثال (طبقه‌بندی نقد فیلم‌ها) : نمودار اتلاف اعتبارسنجی

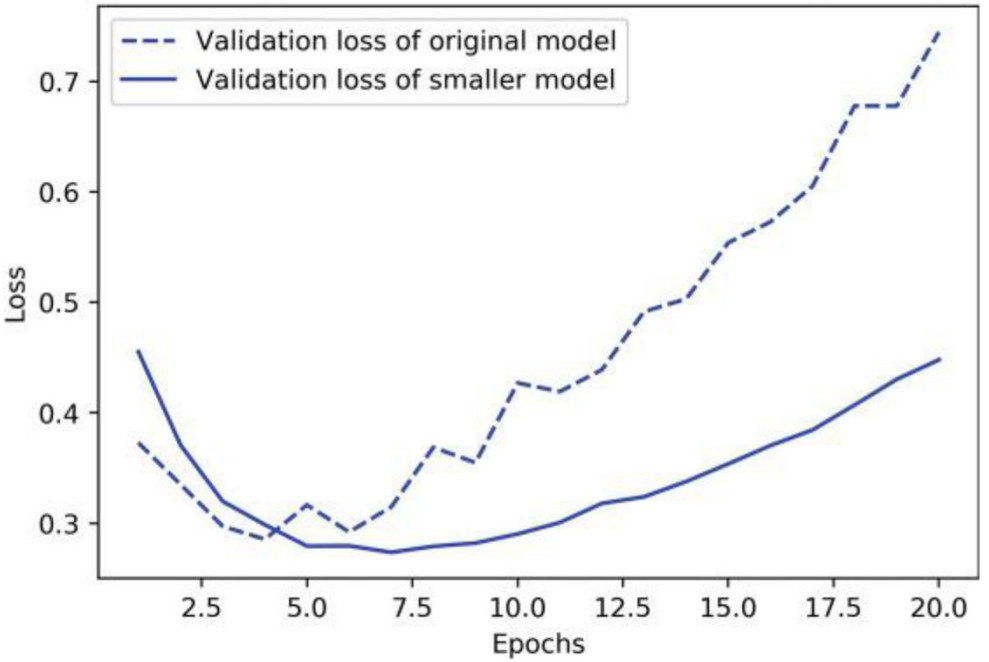


Figure 5.17 Original model vs. smaller model on IMDB review classification

Effect of model capacity on validation loss: trying a smaller model

شبکه‌ی کوچک‌تر، دیرتر از شبکه‌ی اصلی شروع به بیش‌برازش می‌کند (پس از ۶ اپک در مقابل ۴ اپک).
و با شروع بیش‌برازش، کارآیی آن کندتر تنزل می‌یابد.

تکنیک‌های رگولاریزاسیون

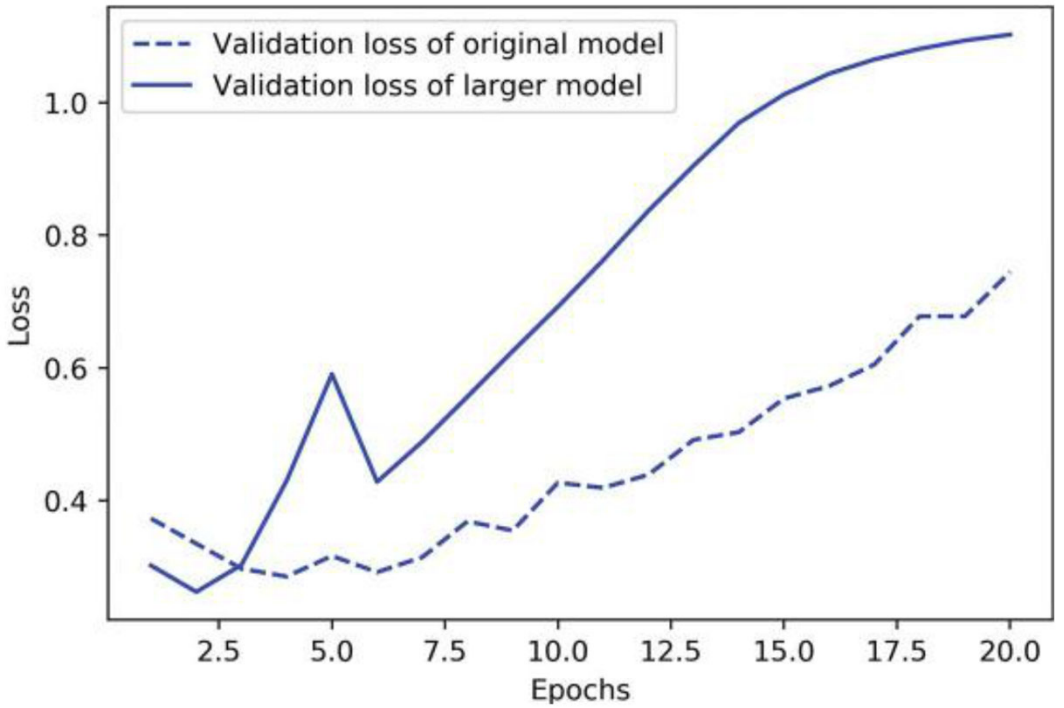
کاهش اندازه‌ی شبکه : مثال (طبقه‌بندی نقد فیلم‌ها) : افزایش ظرفیت شبکه

Listing 5.12 Version of the model with higher capacity

```
model = keras.Sequential([
    layers.Dense(512, activation="relu"),
    layers.Dense(512, activation="relu"),
    layers.Dense(1, activation="sigmoid")
])
model.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
history_larger_model = model.fit(
    train_data, train_labels,
    epochs=20, batch_size=512, validation_split=0.4)
```

تکنیک‌های رگولاریزاسیون

کاهش اندازه‌ی شبکه : مثال (طبقه‌بندی نقد فیلم‌ها) : افزایش ظرفیت شبکه : نمودار اتلاف اعتبارسنجی

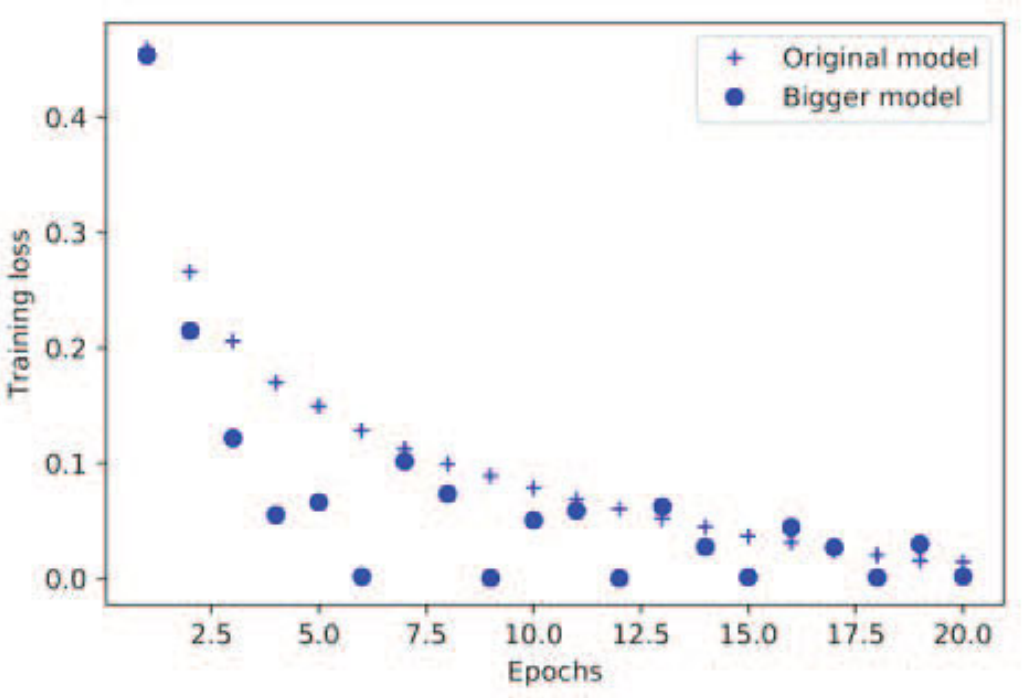


Effect of model capacity on validation loss: trying a bigger model

شبکه‌ی بزرگ‌تر، تقریباً بلافاصله پس از اپک اول شروع به بیش‌برازش می‌کند
و بیش‌برازش آن بسیار شدیدتر است.
سیگنال اتلاف اعتبارسنجی نیز نویزی‌تر است.

تکنیک‌های رگولاریزاسیون

کاهش اندازه‌ی شبکه : مثال (طبقه‌بندی نقد فیلم‌ها) : نمودار اتلاف داده‌های آموزشی



Effect of model capacity on training loss: trying a bigger model

شبکه‌ی بزرگ‌تر در میزان اتلاف داده‌های آموزشی خود سریعاً به نزدیک صفر دست یافته است.
هرچه شبکه ظرفیت بیشتری داشته باشد، داده‌های آموزشی را سریع‌تر می‌تواند مدل کند (منجر به اتلاف کمتر)
اما به بیش‌برازش بیشتر مشکوک می‌شود (در اثر اختلاف بزرگ بین اتلاف‌های آموزشی و اعتبارسنجی).

تکنیک‌های رگولاریزاسیون

اصل تیغه‌ی اوخامی

OCKHAM'S RAZOR

اصل تیغه‌ی اوخامی
Ockham's Razor

ساده‌ترین مدلی که داده‌ها را توضیح می‌دهد، بیابید.

Find the simplest model that explains the data.

هرچه مدل پیچیده‌تر باشد، امکان خطا بیشتر می‌شود.

وقتی دو توضیح برای چیزی داده شده باشد، توضیحی که ساده‌تر است شانس بیشتری برای درست بودن دارد.
(توضیحی که فرضیات کمتری را در نظر می‌گیرد.)

استفاده از این ایده در مدل‌هایی که با شبکه‌های عصبی یاد گرفته می‌شوند:
با داشتن مقداری داده‌های آموزشی و یک معماری شبکه،
چندین مجموعه از مقادیر وزن‌ها (چندین مدل‌ها) میتوانند داده‌ها را توضیح دهند.
مدل‌های ساده‌تر کمتر از مدل‌های پیچیده شانس گرفتار شدن در بیش‌برازش را دارند.

تکنیک‌های رگولاریزاسیون

افزودن رگولاریزاسیون وزنی

ADDING WEIGHT REGULARIZATION

مدل‌های ساده‌تر کمتر از مدل‌های پیچیده شانس گرفتار شدن در بیش‌برازش را دارند.
مدل ساده‌تر در این مضمون، مدلی است که در آن توزیع مقادیر پارامتر آنتروپی کمتری داشته باشد.
(یا مدلی که تعداد پارامتر کمتری داشته باشد.)

پس یک راه متداول برای کاهش بیش‌برازش، قرار دادن قیدهایی بر روی پیچیدگی یک شبکه است:
مجبور کردن وزن‌های شبکه به اینکه تنها مقادیر کوچک را بپذیرند
⇐ توزیع مقادیر وزن‌ها منظم‌تر می‌شود (more regular): **رگولاریزاسیون وزنی**

رگولاریزاسیون وزنی *Weight Regularization*	اضافه کردن یک مقدار هزینه مرتبط با داشتن وزن‌های بزرگ به تابع اتلاف شبکه

انواع متداول رگولاریزاسیون وزنی	○ **رگولاریزاسیون ال۱** (L1 regularization) [the L1 norm of the weights]: هزینه‌ی اضافه شده به اتلاف متناسب با قدرمطلق مقدار ضرایب وزنی است. ○ **رگولاریزاسیون ال۲** (L2 regularization) [the L2 norm of the weights]: هزینه‌ی اضافه شده به اتلاف متناسب با مجذور مقدار ضرایب وزنی است.

Regularization: Add term to loss

$$L = \frac{1}{N}\sum_{i=1}^{N}\sum_{j\neq y_i}\max(0, f(x_i; W)_j - f(x_i; W)_{y_i} + 1) + \boxed{\lambda R(W)}$$

In common use:

L2 regularization $R(W) = \sum_k \sum_l W_{k,l}^2$ (Weight decay)

L1 regularization $R(W) = \sum_k \sum_l |W_{k,l}|$

Elastic net (L1 + L2) $R(W) = \sum_k \sum_l \beta W_{k,l}^2 + |W_{k,l}|$

Ref: http://cs231n.stanford.edu/

تکنیک‌های رگولاریزاسیون

افزودن رگولاریزاسیون وزنی در کراس

In **Keras**, weight regularization is added
by passing weight regularizer instances to layers as keyword arguments.

Let's add L2 weight regularization to the movie-review classification network.

Listing 5.13 Adding L2 weight regularization to the model

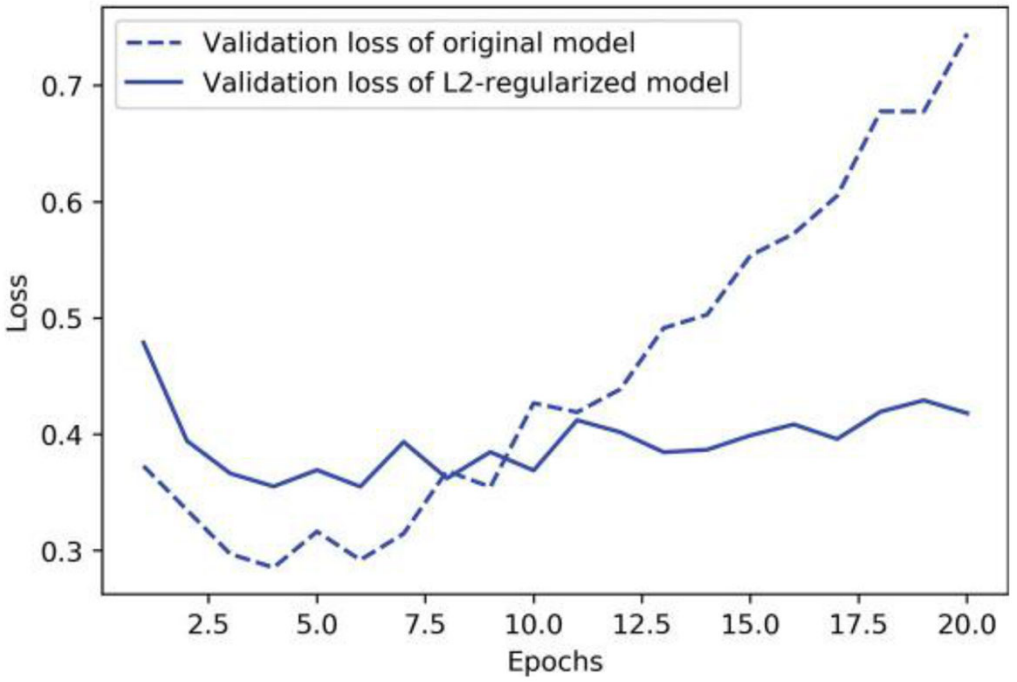
```
from tensorflow.keras import regularizers
model = keras.Sequential([
    layers.Dense(16,
                 kernel_regularizer=regularizers.l2(0.002),
                 activation="relu"),
    layers.Dense(16,
                 kernel_regularizer=regularizers.l2(0.002),
                 activation="relu"),
    layers.Dense(1, activation="sigmoid")
])
model.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
history_l2_reg = model.fit(
    train_data, train_labels,
    epochs=20, batch_size=512, validation_split=0.4)
```

`l2(0.001)` means every coefficient in the weight matrix of the layer
will add `0.001 * weight_coefficient_value` to the total loss of the network.

تذکر: جریمه‌ی رگولاریزاسیون تنها در زمان آموزش اضافه می‌شود،
برای همین میزان اتلاف شبکه در زمان آموزش بسیار بزرگ‌تر از زمان تست است.

تکنیک‌های رگولاریزاسیون

افزودن رگولاریزاسیون وزنی در کراس: نمودار اتلاف اعتبارسنجی



Effect of L2 weight regularization on validation loss

مدل دارای رگولاریزاسیون L2 نسبت به مدل مرجع در برابر بیش‌برازش بیشتر مقاومت می‌کند.
اگرچه هر دو مدل تعداد یکسانی پارامتر دارند.

تکنیک‌های رگولاریزاسیون

افزودن رگولاریزاسیون وزنی در کراس: گزینه‌ها

As an alternative to L2 regularization,
you can use one of the following **Keras** weight regularizers.

Listing 5.14 Different weight regularizers available in Keras

```
from tensorflow.keras import regularizers
regularizers.l1(0.001)
regularizers.l1_l2(l1=0.001, l2=0.001)
```

L1 regularization

Simultaneous L1 and L2 regularization

تکنیک‌های رگولاریزاسیون

اضافه کردن برون‌اندازی

ADDING DROPOUT

برون‌اندازی/Dropout یکی از مؤثرترین و متداول‌ترین تکنیک‌های رگولاریزاسیون برای شبکه‌های عصبی است.
(توسعه یافته توسط هینتون و دانشجویان او در دانشگاه تورنتو)

برون‌اندازی *Dropout*	اعمال Dropout به یک لایه، عبارت است از حذف تصادفی (صفر کردن) تعدادی از ویژگی‌های خروجی آن لایه در هنگام آموزش

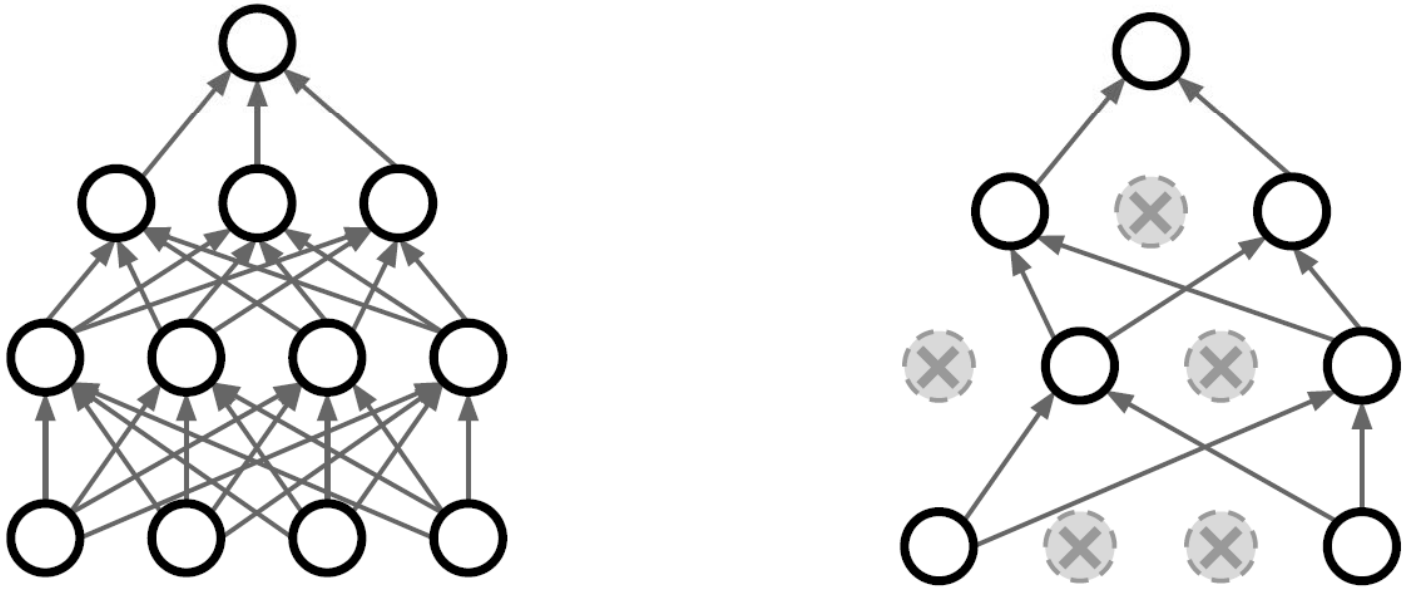
نرخ برون‌اندازی *Dropout Rate*	کسر تعداد ویژگی‌هایی که صفر شده و خارج می‌شوند. این مقدار معمولاً بین 0.2 و 0.5 است.

در زمان تست، هیچ واحدی dropout نمی‌شود؛
در عوض، خروجی لایه‌ها با فاکتوری برابر با نرخ dropout مقیاس می‌شوند.
(برای متعادل‌سازی بر اساس این واقعیت که تعداد واحدهای بیشتری نسبت به زمان آموزش فعال است.)

Regularization: Dropout

In each forward pass, randomly set some neurons to zero

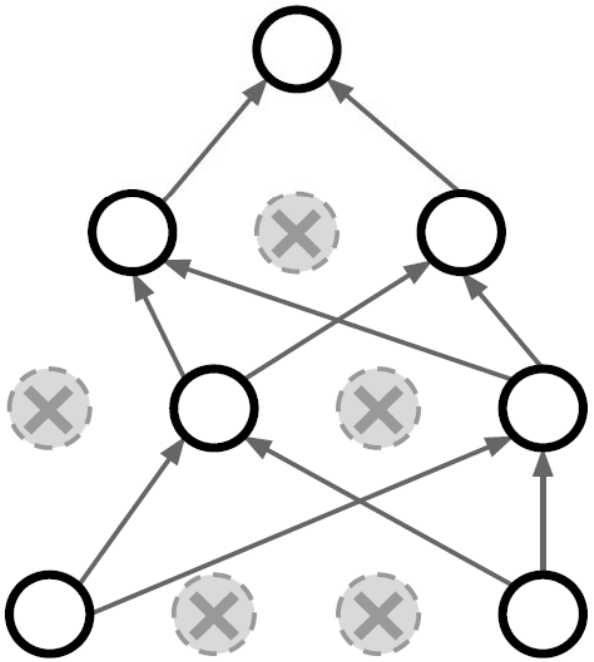
Probability of dropping is a hyperparameter; 0.5 is common



Ref: http://cs231n.stanford.edu/

Regularization: Dropout

How can this possibly be a good idea?

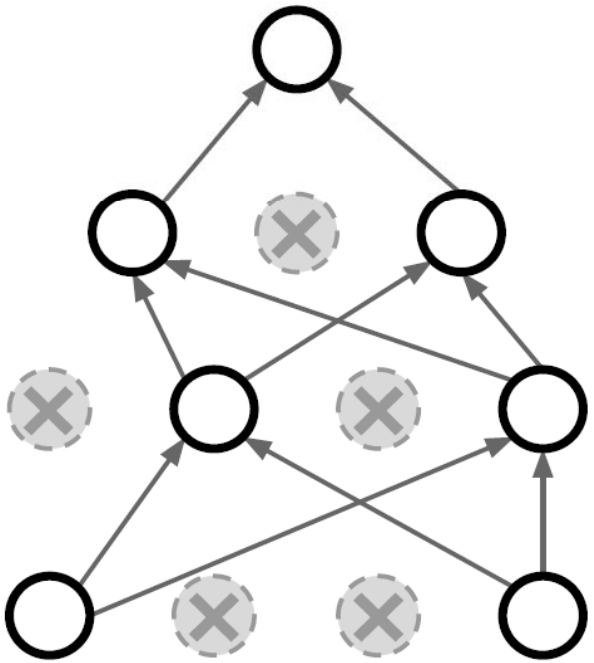


Forces the network to have a redundant representation;
Prevents co-adaptation of features

Ref: http://cs231n.stanford.edu/

Regularization: Dropout

How can this possibly be a good idea?



Another interpretation:

Dropout is training a large **ensemble** of models (that share parameters).

Each binary mask is one model

An FC layer with 4096 units has
$2^{4096} \sim 10^{1233}$ possible masks!
Only $\sim 10^{82}$ atoms in the universe...

Ref: http://cs231n.stanford.edu/

Dropout: Test time

Dropout makes our output random!

Output (label) Input (image)

$$y = f_W(x, z)$$

Random mask

Want to "average out" the randomness at test-time

$$y = f(x) = E_z\big[f(x, z)\big] = \int p(z) f(x, z) dz$$

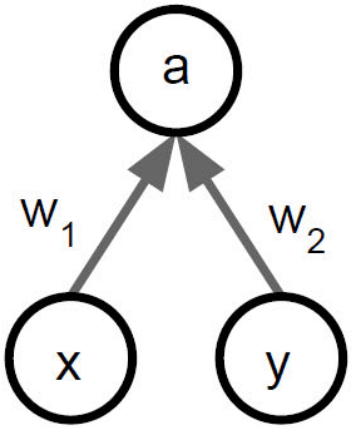
But this integral seems hard …

Ref: http://cs231n.stanford.edu/

Dropout: Test time

Want to approximate the integral

$$y = f(x) = E_z\big[f(x,z)\big] = \int p(z) f(x,z) dz$$



Consider a single neuron.

At test time we have: $E\big[a\big] = w_1 x + w_2 y$

During training we have:

$$\begin{aligned} E[a] =& \frac{1}{4}(w_1 x + w_2 y) + \frac{1}{4}(w_1 x + 0y) \\ &+ \frac{1}{4}(0x + 0y) + \frac{1}{4}(0x + w_2 y) \\ =& \frac{1}{2}(w_1 x + w_2 y) \end{aligned}$$

At test time, **multiply** by dropout probability

Ref: http://cs231n.stanford.edu/

Dropout: Test time

```
def predict(X):
  # ensembled forward pass
  H1 = np.maximum(0, np.dot(W1, X) + b1) * p # NOTE: scale the activations
  H2 = np.maximum(0, np.dot(W2, H1) + b2) * p # NOTE: scale the activations
  out = np.dot(W3, H2) + b3
```

At test time all neurons are active always
=> We must scale the activations so that for each neuron:
output at test time = expected output at training time

Ref: http://cs231n.stanford.edu/

تکنیک‌های رگولاریزاسیون

اضافه کردن برون‌اندازی

Consider a Numpy matrix containing the output of a layer, layer_output,
of shape `(batch_size, features)`.
At training time, we zero out at random a fraction of the values in the matrix:

```
layer_output *= np.random.randint(0, high=2, size=layer_output.shape)
                        # At training time, drops out 50% of the units in the output
```

At test time, we scale down the output by the dropout rate. Here, we scale by 0.5
(because we previously dropped half the units):

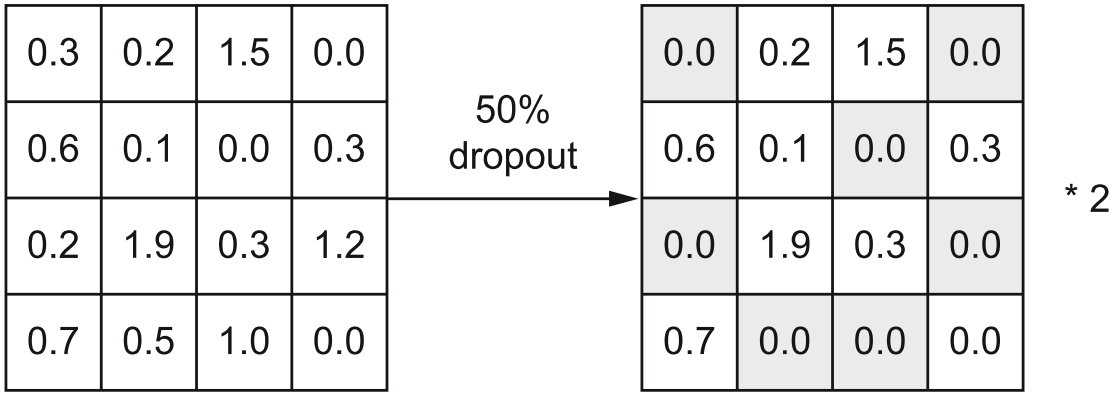
```
layer_output *= 0.5                              # At test time
```

تکنیک‌های رگولاریزاسیون

اضافه کردن برون‌اندازی

Note that this process can be implemented by doing both operations at training time and leaving the output unchanged at test time, which is often the way it's implemented in practice:

```
layer_output *= np.random.randint(0, high=2, size=layer_output.shape)
                                                    # At training time
layer_output /= 0.5
                    # Note that we're scaling up rather scaling down in this case.
```



Dropout applied to an activation matrix at training time,
with rescaling happening during training.
At test time, the activation matrix is unchanged.

تکنیک‌های رگولاریزاسیون

اضافه کردن برون‌اندازی: منشأ ایده

تکنیک برون‌اندازی/ Dropout ممکن است عجیب و دلبخواهی به‌نظر برسید.
چرا این تکنیک بیش‌برازش را کاهش می‌دهد؟
هینتون بیان می‌کند که این روش را
غیر از سایر موارد، از یک مکانیسم پیشگیری از کلاهبرداری در بانک‌ها ایده گرفته است:

«من به بانک رفته بودم، تحویل‌دارها جای خود را با یکدیگر عوض کرده بودند و من دلیل این کار را از یکی از آنها پرسیدم. او گفت که نمی‌داند، اما آنها زیاد جای خود را تغییر می‌دهند.
من با خودم فکر کردم که دلیل آن حتماً این است که پیشگیری موفقیت‌آمیز از کلاهبرداری از بانک نیازمند همکاری بین کارمندان است. این نکته باعث شد تا متوجه بشوم تغییر مکان تصادفی یک زیرمجموعه‌ی متفاوت از نرون‌ها در هر مثال می‌تواند مانع از توطئه و دسیسه شود و بنابراین بیش‌برازش را کاهش می‌دهد.»

ایده و مفهوم اصلی:

وارد کردن نویز در مقادیر خروجی یک لایه،
می‌تواند سبب شود الگوهای اتفاقی که چشمگیر نیستند (توطئه‌ها) بشکنند
اما اگر هیچ تداخل و نویزی وجود نداشته باشد، شبکه شروع به حفظ کردن داده‌ها می‌کند (بیش‌برازش).

تکنیک‌های رگولاریزاسیون

اضافه کردن Dropout در کراس

In **Keras**, you can introduce dropout in a network via the Dropout layer, which is applied to the output of the layer right before it

```
layers.Dropout(0.5)
```

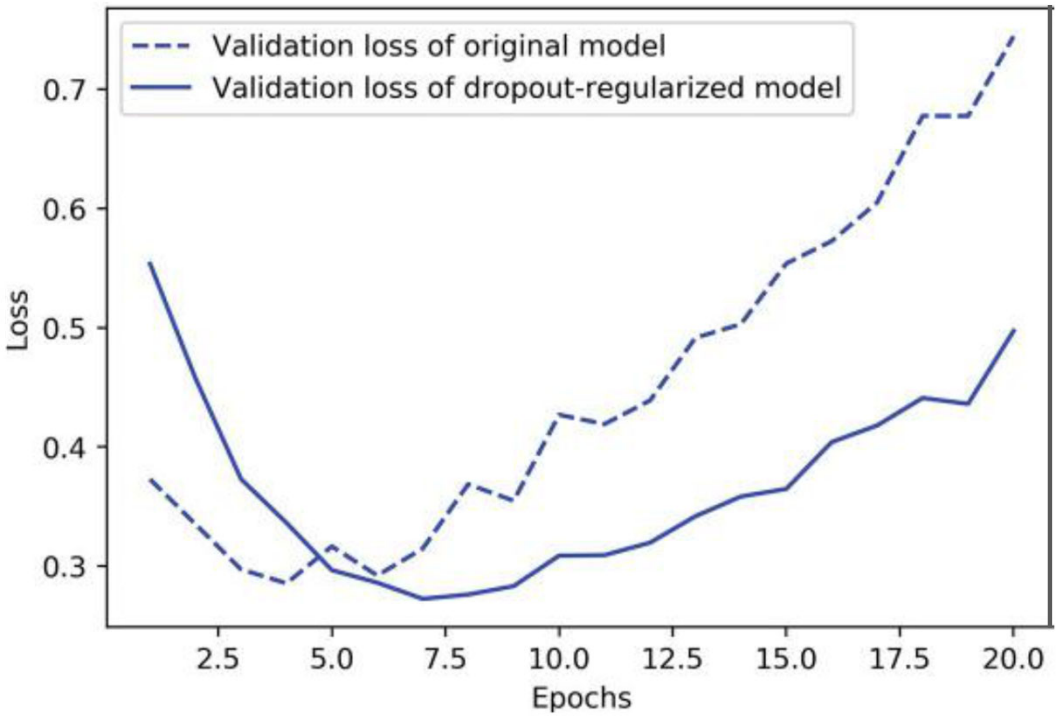
Let's add two Dropout layers in the IMDB network to see how well they do at reducing overfitting.

Listing 5.15 Adding dropout to the IMDB model

```
model = keras.Sequential([
    layers.Dense(16, activation="relu"),
    layers.Dropout(0.5),
    layers.Dense(16, activation="relu"),
    layers.Dropout(0.5),
    layers.Dense(1, activation="sigmoid")
])
model.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
history_dropout = model.fit(
    train_data, train_labels,
    epochs=20, batch_size=512, validation_split=0.4)
```

تکنیک‌های رگولاریزاسیون

اضافه کردن Dropout : نمودار اعتبارسنجی



Effect of dropout on validation loss

این نمودار بیانگر پیشرفت آشکاری نسبت به شبکه‌ی مرجع است.

جمع‌بندی: اجتناب از بیش‌برازش در شبکه‌های عصبی: تعمیم‌پذیری بیشتر

RECAP: TO PREVENT OVERFITTING IN NEURAL NETWORKS: MORE GENERALIZATION

متداول‌ترین روش‌ها برای جلوگیری از بروز بیش‌برازش در شبکه‌های عصبی:

یادگیری عمیق

مبانی یادگیری ماشینی

۳

گردش کار جامع در یادگیری ماشینی

گردش کار جامع در یادگیری ماشینی

The Universal Workflow of Machine Learning

گردش کار جامع در یادگیری ماشینی

تعریف مسئله و گردآوری مجموعه‌داده

DEFINING THE PROBLEM AND ASSEMBLING A DATASET

۱ تعریف مسئله و گردآوری مجموعه‌داده
Defining the problem and assembling a dataset

تعریف مسئله

- داده‌های ورودی؟ چه چیزی باید پیش‌بینی شود؟
- ورودی؟ خروجی؟
- نوع مسئله؟ (طبقه‌بندی دودویی، رگرسیون، خوشه‌بندی، تولید، ...)

شناسایی نوع مسئله، در انتخاب مدل، تابع اتلاف و ... ما را راهنمایی می‌کند.

فرضیات اساسی

- فرض می‌شود که می‌توان خروجی‌ها را بر اساس ورودی‌های آنها پیش‌بینی کرد. [وجود رابطه‌ی تابعی بین ورودی‌ها و خروجی‌ها]
- فرض می‌شود که داده‌های موجود حاوی اطلاعات کافی برای یادگیری رابطه‌ی بین ورودی‌ها و خروجی‌ها هستند. [میزان داده‌ی کافی]

گردش کار جامع در یادگیری ماشینی

تعریف مسئله و گردآوری مجموعه‌داده: مسائل حل‌ناپذیر

UNSOLVABLE PROBLEMS

مسائل غیرایستان
Nonstationary Problems

مسائلی که هدف از آنها کشف تابعی است که ضابطه‌ی آن با زمان تغییر می‌کند.

مثال: مسئله‌ی سیستم توصیه‌گر لباس
توصیه متناسب با زمان سال متغیر است،
پس داده‌های آموزشی باید حاوی ویژگی زمان سال باشند.

گردش کار جامع در یادگیری ماشینی

تعریف مسئله و گردآوری مجموعه‌داده : محدودیت‌ها

یادگیری ماشینی تنها می‌تواند
برای حفظ کردن الگوهایی که در مجموعه داده‌های آموزشی حاضر است
استفاده شود.

«چیزی را می‌توانیم بازشناسی کنیم که قبلاً آن را دیده باشیم»

استفاده از یادگیری ماشینی آموزش‌دیده بر روی داده‌های گذشته برای پیش‌بینی آینده،
این فرض را در خود دارد که:
«آینده مانند گذشته رفتار می‌کند»
که خیلی اوقات برقرار نیست!

گردش کار جامع در یادگیری ماشینی

انتخاب یک معیار موفقیت

CHOOSING A MEASURE OF SUCCESS

۲ انتخاب یک معیار موفقیت
Choosing a measure of success

برای اینکه بتوان چیزی را کنترل کرد، باید بتوان آن را مشاهده کرد.
برای دستیابی به موفقیت باید آن را تعریف کنیم: دقت؟ یادآوری؟ نرخ حفظ مشتری؟ ...

معیار موفقیت، راهنمایی برای انتخاب یک تابع هزینه خواهد بود.

- برای مسائل طبقه‌بندی متوازن (احتمال کلاس‌ها مساوی)، **صحت (Accuracy)** و **سطح زیر منحنی ROC** یک معیار متداول است (ROC AUC).
- برای مسائل طبقه‌بندی نامتوازن (احتمال کلاس‌ها نامساوی)، **دقت (Precision)** و **یادآوری (Recall)** متداول است.
- برای مسائل رتبه‌بندی یا طبقه‌بندی چندبرچسبی، **میانگین دقت متوسط (MAP)** متداول است.

تعریف معیار موفقیت اختصاصی برای کاربردهای گوناگون، متداول است.

گردش کار جامع در یادگیری ماشینی

تصمیم‌گیری در مورد پروتکل ارزیابی

DECIDING ON AN EVALUATION PROTOCOL

۳ تصمیم‌گیری در مورد پروتکل ارزیابی
Deciding on an evaluation protocol

وقتی که هدف مشخص شد، باید شیوه‌ای برای سنجش **پیشرفت فرآیند یادگیری** مشخص شود.

سه پروتکل متداول برای ارزیابی داریم:

اعتبارسنجی ساده نگه‌داشت–برون‌گذاشت *Simple Hold-Out Validation*	روش مناسب برای زمانی که حجم فراوانی داده در اختیار داریم. (در بیشتر موارد این روش به‌اندازه‌ی کافی خوب است)
اعتبارسنجی *K-fold* *K-fold Validation*	انتخاب درست برای مواقعی که نمونه‌های بسیار اندکی در اختیار داریم و روش ساده اطمینان‌بخش نیست.
اعتبارسنجی *K-fold* تکراری با بُر زدن *Iterated K-fold Validation with Shuffling*	برای اجرای ارزیابی بسیار دقیق مدل در زمانی که داده‌های کمی در دسترس است.

گردش کار جامع در یادگیری ماشینی

آماده‌سازی داده‌ها

PREPARING THE DATA

۴	آماده‌سازی داده‌ها *Preparing the data*

ابتدا باید داده‌ها را به‌گونه‌ای که بتوانند وارد یک مدل یادگیری ماشینی (در اینجا شبکه‌ی عصبی) شوند، فرمت‌بندی کنید.

فرمت‌بندی برای آماده‌سازی داده‌ها

- داده‌ها باید در قالب تانسورها فرمت‌بندی شوند.
- مقادیر موجود در تانسورها باید در قالب اعداد کوچک مقیاس شوند. (بازه‌ی [0,1] یا [-1,1])
- نرمال‌سازی مستقل ویژگی‌ها برای داده‌های ناهمگن
- انجام مهندسی ویژگی‌ها به‌خصوص برای مسائلی با داده‌های کوچک

زمانی که تانسورهای داده‌های ورودی و تارگت آماده شدند، می‌توانیم آموزش مدل را شروع کنیم.

گردش کار جامع در یادگیری ماشینی

توسعه‌ی یک مدل که بهتر از یک مدل پایه عمل کند

DEVELOPING A MODEL THAT DOES BETTER THAN A BASELINE

۵ توسعه‌ی یک مدل که بهتر از یک مدل پایه عمل کند
Developing a model that does better than a baseline

هدف این مرحله، توسعه‌ی مدلی است که به **قدرت آماری** دست پیدا کند.

قدرت آماری
Statistical Power

مدلی دارای قدرت آماری است که
قادر به شکست دادن یک مدل پایه‌ی تصادفی باشد.

برای مثال: مدل پایه تصادفی برای مسئله‌ی طبقه‌بندی دو کلاسی دارای دقت $0.5 = ½$ است.
مدل پایه تصادفی برای مسئله‌ی طبقه‌بندی n–کلاسی دارای دقت $1/n$ است.

دستیابی به مدلی با قدرت آماری، همیشه ممکن نیست.
اگر پس از آزمایش چندین معماری مستدل نتوانیم مدل پایه‌ی تصادفی را شکست دهیم،
ممکن است به این دلیل باشد که پاسخ پرسشی که به‌دنبال آن هستیم در داده‌های ورودی وجود ندارد.
یا ممکن است فرضیات اساسی (وجود رابطه‌ی تابعی / داده‌ی کافی) نادرست باشند.

گردش کار جامع در یادگیری ماشینی

توسعه‌ی یک مدل که بهتر از یک مدل پایه عمل کند: پارامترها

با فرض درست بودن فرضیات اولیه،
باید در مورد سه انتخاب کلیدی برای ساختن اولین مدل کاری خود تصمیم‌گیری کنید:

سه انتخاب کلیدی برای ساختن اولین مدل کاری

- تابع فعالیت لایه‌ی آخر [بر اساس نوع خروجی شبکه]
- تابع اتلاف [لزوم مطابقت با مسئله‌ی مورد نظر]
- پیکربندی بهینه‌ساز (الگوریتم بهینه‌سازی، نرخ یادگیری آن، ...)

با توجه به انتخاب تابع اتلاف، همواره امکان بهینه‌سازی مستقیم برای معیارهای موفقیت مسئله وجود ندارد.
در برخی موارد هیچ راه آسانی برای تبدیل یک معیار موفقیت به یک تابع هزینه وجود ندارد.
تابع اتلاف باید برای یک دسته‌ی کوچک از داده‌ها قابل محاسبه باشد.
تابع اتلاف باید حتی برای یک نقطه هم قابل محاسبه باشد.
تابع اتلاف باید مشتق‌پذیر باشد تا بتوان از آن در الگوریتم پس‌انتشار خطا استفاده کرد.
[برای مثال معیار ROC AUC مشتق‌پذیر نیست
و برای همین در مسائل طبقه‌بندی از معیارهای نماینده‌ی آن مانند آنتروپی متقابل استفاده می‌شود.]

گردش کار جامع در یادگیری ماشینی

توسعه‌ی یک مدل که بهتر از یک مدل پایه عمل کند: انتخاب تابع فعالیت و تابع اتلاف برای مسائل گوناگون

Problem type	Last-layer activation	Loss function
Binary classification	`sigmoid`	`binary_crossentropy`
Multiclass, single-label classification	`softmax`	`categorical_crossentropy`
Multiclass, multilabel classification	`sigmoid`	`binary_crossentropy`
Regression to arbitrary values	None	`mse`
Regression to values between 0 and 1	`sigmoid`	`mse` or `binary_crossentropy`

گردش کار جامع در یادگیری ماشینی

افزایش مقیاس: توسعه‌ی یک مدل که بیش‌برازش کند

SCALING UP: DEVELOPING A MODEL THAT OVERFITS

۶ افزایش مقیاس: توسعه‌ی یک مدل که بیش‌برازش کند
Scaling up: developing a model that overfits

وقتی به مدلی رسیدیم که دارای قدرت آماری است، این پرسش مطرح می‌شود که:
آیا این مدل به‌اندازه‌ی کافی قدرت‌مند است؟
[تعداد لایه‌ها و واحدها کافی است؟ ...]

کشمکش اساسی در یادگیری ماشینی بین سازگاری و تعمیم‌دهی است.
مدل ایده‌آل، مدلی است که درست در مرز میان **کم‌برازش** و **بیش‌برازش** قرار بگیرد:
مرز میان کم‌ظرفیتی و بیش‌ظرفیتی؛
برای پیدا کردن این مرز ابتدا باید از آن عبور کنیم.

برای تعیین بزرگی مدل مورد نیاز، باید مدلی ایجاد کرد که دچار بیش‌برازش شود:

- لایه‌ها را اضافه می‌کنیم.
- لایه‌ها را بزرگ‌تر می‌کنیم.
- فرآیند آموزش را طی اپک‌های بیشتری تکرار می‌کنیم.

اتلاف روی داده‌های آموزشی و اعتبارسنجی را تحت نظارت قرار می‌دهیم.
هنگامی که کارآیی مدل روی داده‌های اعتبارسنجی شروع به افت کرد، به بیش‌برازش رسیده‌ایم.

گردش کار جامع در یادگیری ماشینی

رگولاریزاسیون مدل و تنظیم هایپرپارامترها

REGULARIZING THE MODEL AND TUNING YOUR HYPERPARAMETERS

۷ رگولاریزاسیون مدل و تنظیم هایپرپارامترها
Regularizing the model and tuning your hyperparameters

این مرحله بیشترین زمان را صرف می‌کند:
مدل باید به‌طور مکرر اصلاح شود، تحت آموزش مجدد قرار گیرد و روی داده‌های اعتبارسنجی ارزیابی شود.
مجدداً مدل اصلاح می‌شود و تکرار می‌شود تا زمانی که مدل به بهترین حالت برسد.

موارد قابل امتحان کردن

- اضافه کردن برون‌اندازی / Dropout
- آزمون معماری‌های مختلف: اضافه / حذف لایه‌ها
- اضافه کردن رگولاریزاسیون وزنی
- آزمون هاپیرپارامترهای مختلف (تعداد واحد هر لایه / نرخ یادگیری بهینه‌ساز، ...)
- مهندسی ویژگی‌ها: اضافه کردن ویژگی جدید، حذف ویژگی غیرمفید

نکته‌ی مهم: هر زمان که فیدبک فرآیند اعتبارسنجی برای تنظیم مدل استفاده می‌شود،
مقداری نشت اطلاعات در مورد فرآیند اعتبارسنجی به درون مدل داریم.
اگر تنها چند بار این اتفاق رخ دهد، خطری ندارد،
اما اگر به‌طور منظم و به‌دفعات رخ بدهد، سرانجام منجر به بیش‌برازش مدل نسبت به فرآیند اعتبارسنجی می‌شود.
در نتیجه فرآیند ارزیابی، قابلیت اطمینان کمتری خواهد داشت.

گردش کار جامع در یادگیری ماشینی

پایان کار

وقتی به یک پیکربندی راضی‌کننده برای مدل رسیدیم،
مدل نهایی را روی تمام داده‌های موجود (آموزشی و اعتبارسنجی) مورد آموزش قرار می‌دهیم
و آن را برای آخرین بار روی مجموعه‌ی آزمایشی ارزیابی می‌کنیم.

اگر مشخص شود که کارآیی در مجموعه‌ی آزمایشی خیلی بدتر از کارآیی در مجموعه‌ی آموزشی است،
ممکن است به این معنا باشد که روال اعتبارسنجی مورد استفاده قابل اطمینان نبوده است،
یا در هنگام تنظیم پارامترهای مدل شروع به بیش‌برازش نسبت به داده‌های اعتبارسنجی کرده است.
در این حالت باید به سراغ یک پروتکل ارزیابی با قابلیت اطمینان بالاتر برویم.

مبانی یادگیری ماشینی

خلاصه

- When you take on a new machine learning project, first **define the problem** at hand:
  - Understand the broader context of what you're setting out to do—what's the end goal and what are the constraints?
  - Collect and annotate a **dataset**; make sure you understand your data in depth.
  - Choose how you'll measure success for your problem—what metrics will you monitor on your validation data?
- Once you understand the problem and you have an appropriate dataset, **develop a model**:
  - Prepare your data.
  - Pick your evaluation protocol: holdout validation? K-fold validation? Which portion of the data should you use for validation?
  - Achieve statistical power: beat a simple baseline.
  - Scale up: develop a model that can overfit.
  - Regularize your model and tune its hyperparameters, based on performance on the validation data. A lot of machine learning research tends to focus only on this step, but keep the big picture in mind.

مبانی یادگیری ماشینی

خلاصه (ادامه)

- When your model is ready and yields good performance on the test data, it's time for deployment:
  - First, make sure you set appropriate expectations with stakeholders.
  - Optimize a final model for inference, and ship a model to the deployment environment of choice—web server, mobile, browser, embedded device, etc.
  - Monitor your model's performance in production, and keep collecting data so you can develop the next generation of the model.

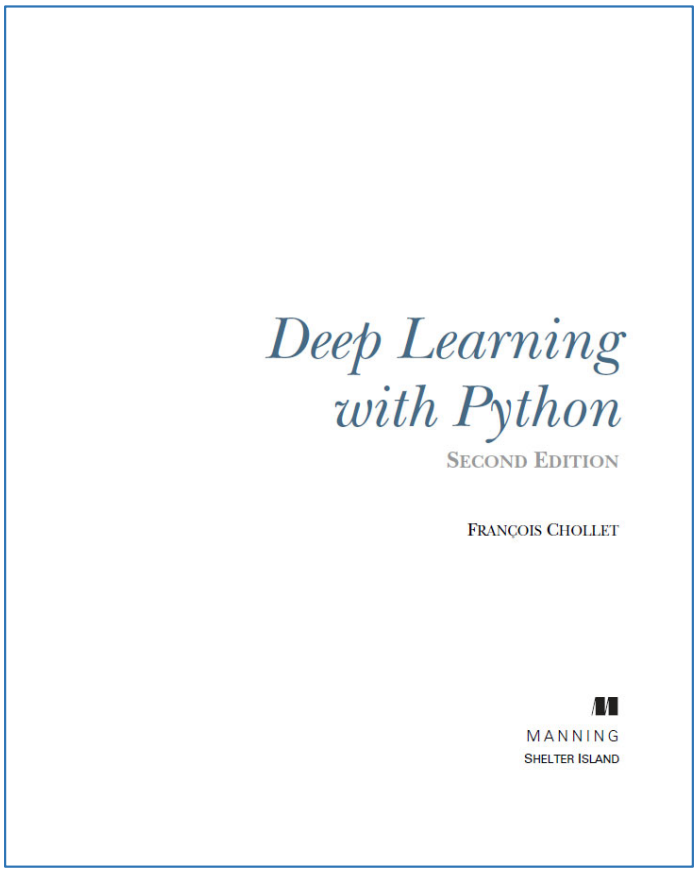
یادگیری عمیق

مبانی یادگیری ماشینی

۴

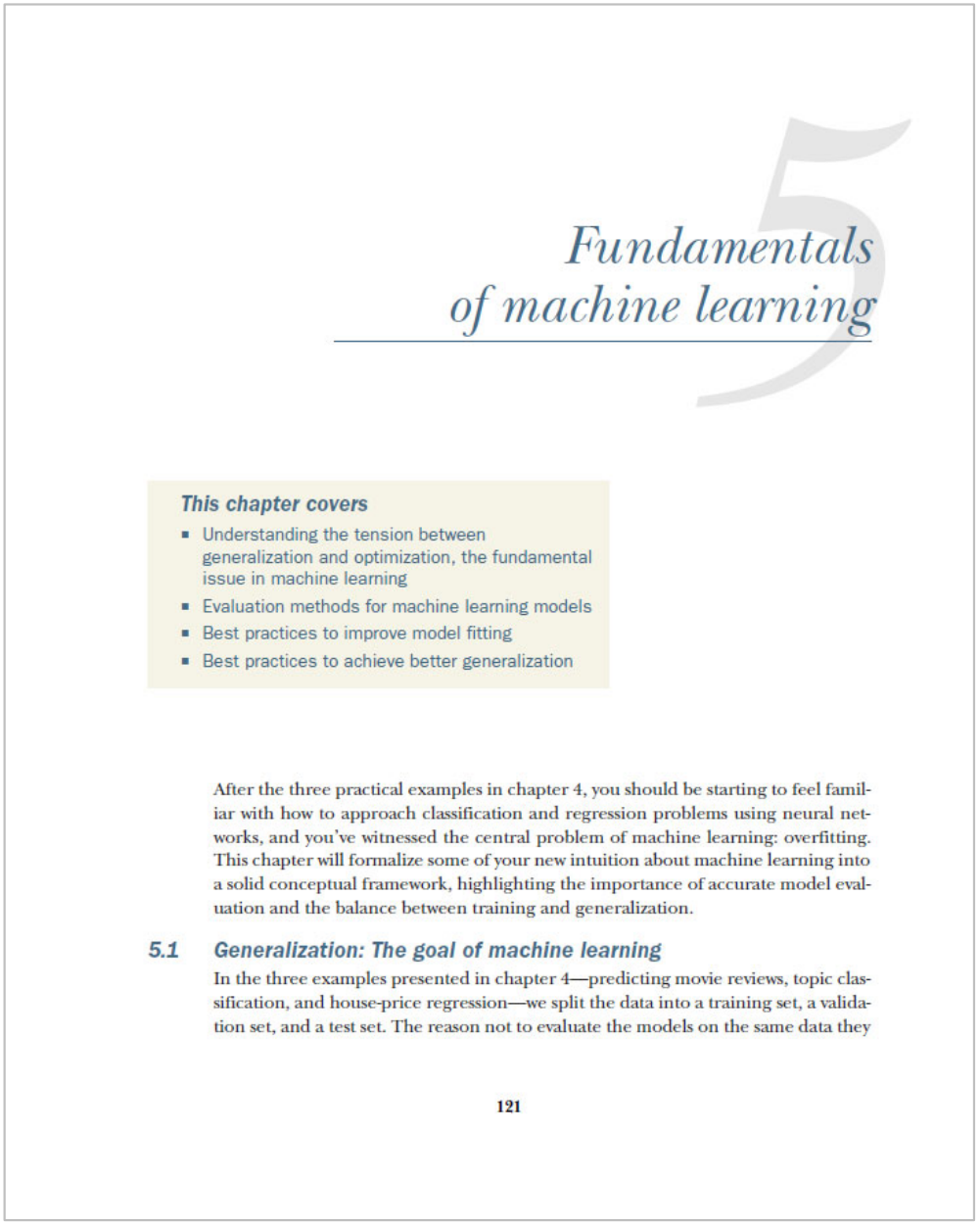
منابع

منبع اصلی



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

Chapter 5



5

Fundamentals of machine learning

This chapter covers

- Understanding the tension between generalization and optimization, the fundamental issue in machine learning
- Evaluation methods for machine learning models
- Best practices to improve model fitting
- Best practices to achieve better generalization

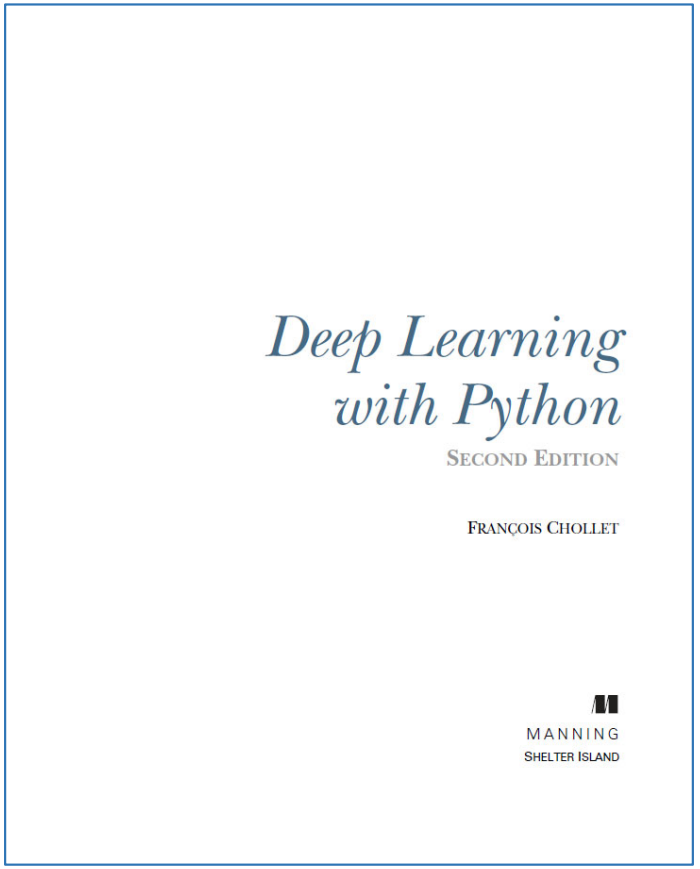
After the three practical examples in chapter 4, you should be starting to feel familiar with how to approach classification and regression problems using neural networks, and you've witnessed the central problem of machine learning: overfitting. This chapter will formalize some of your new intuition about machine learning into a solid conceptual framework, highlighting the importance of accurate model evaluation and the balance between training and generalization.

5.1 Generalization: The goal of machine learning

In the three examples presented in chapter 4—predicting movie reviews, topic classification, and house-price regression—we split the data into a training set, a validation set, and a test set. The reason not to evaluate the models on the same data they

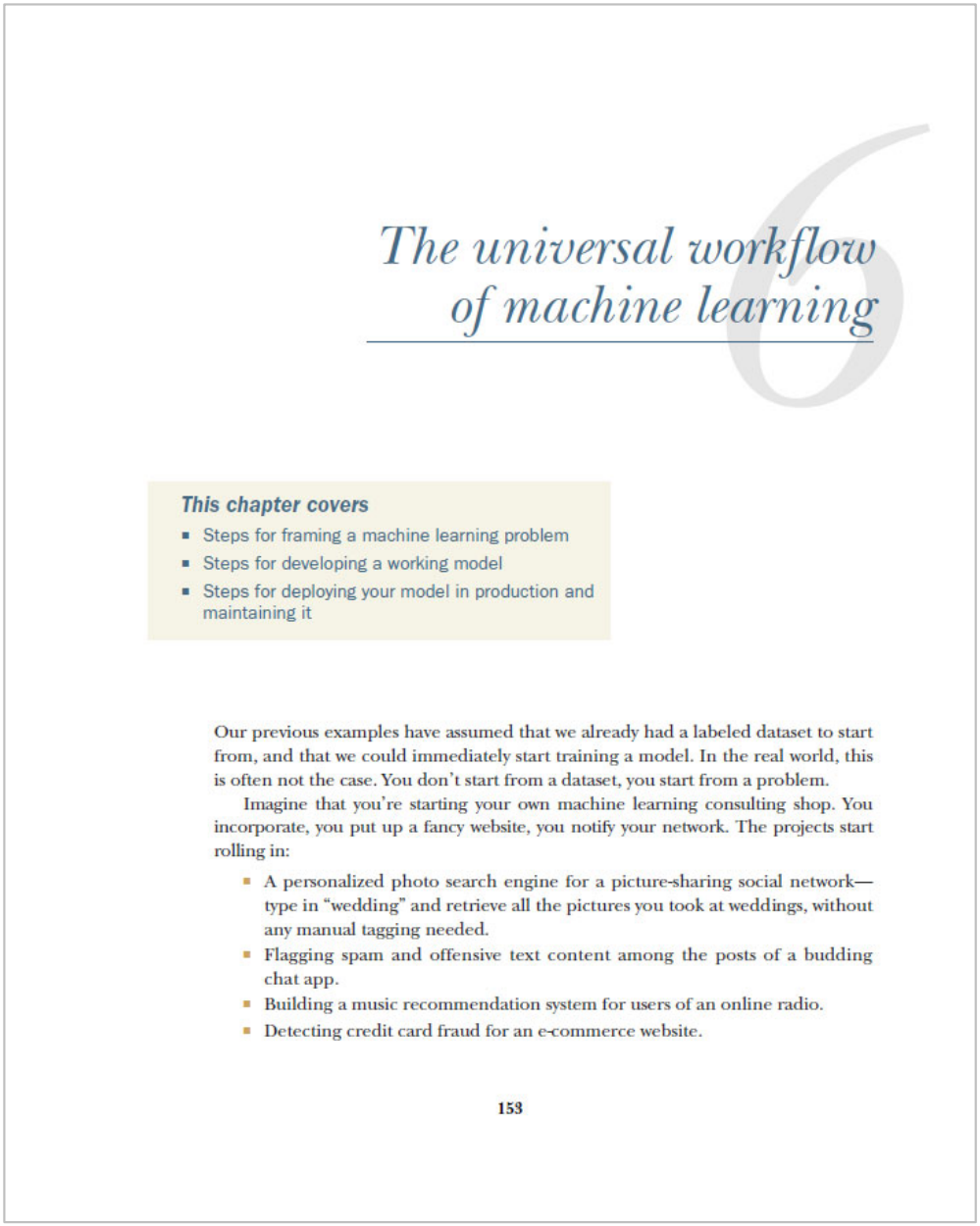
121

منبع اصلی



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

Chapter 6



6

The universal workflow of machine learning

This chapter covers

- Steps for framing a machine learning problem
- Steps for developing a working model
- Steps for deploying your model in production and maintaining it

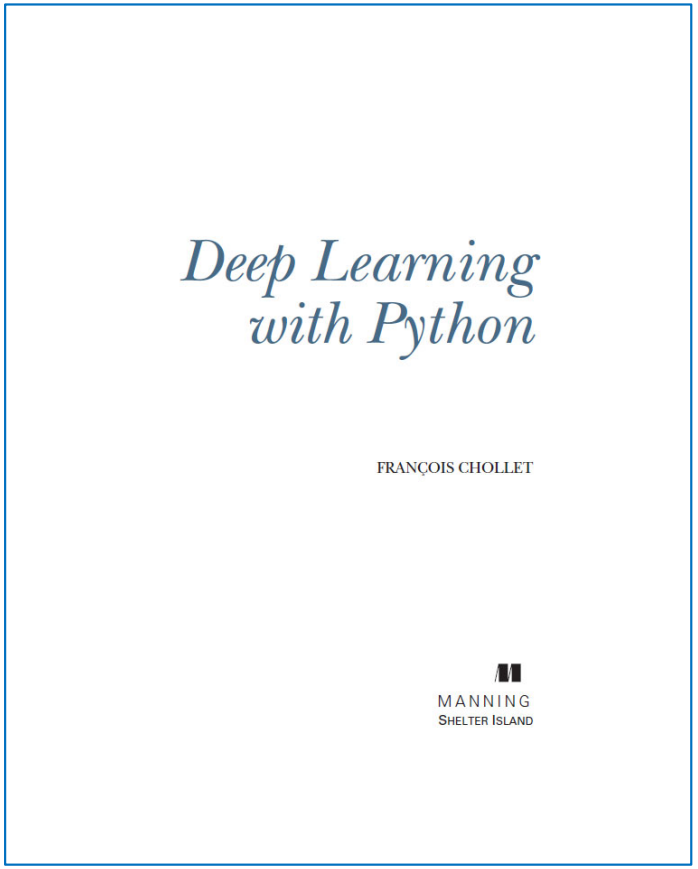
Our previous examples have assumed that we already had a labeled dataset to start from, and that we could immediately start training a model. In the real world, this is often not the case. You don't start from a dataset, you start from a problem.

Imagine that you're starting your own machine learning consulting shop. You incorporate, you put up a fancy website, you notify your network. The projects start rolling in:

- A personalized photo search engine for a picture-sharing social network—type in "wedding" and retrieve all the pictures you took at weddings, without any manual tagging needed.
- Flagging spam and offensive text content among the posts of a budding chat app.
- Building a music recommendation system for users of an online radio.
- Detecting credit card fraud for an e-commerce website.

153

منبع اصلی



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

Chapter 4