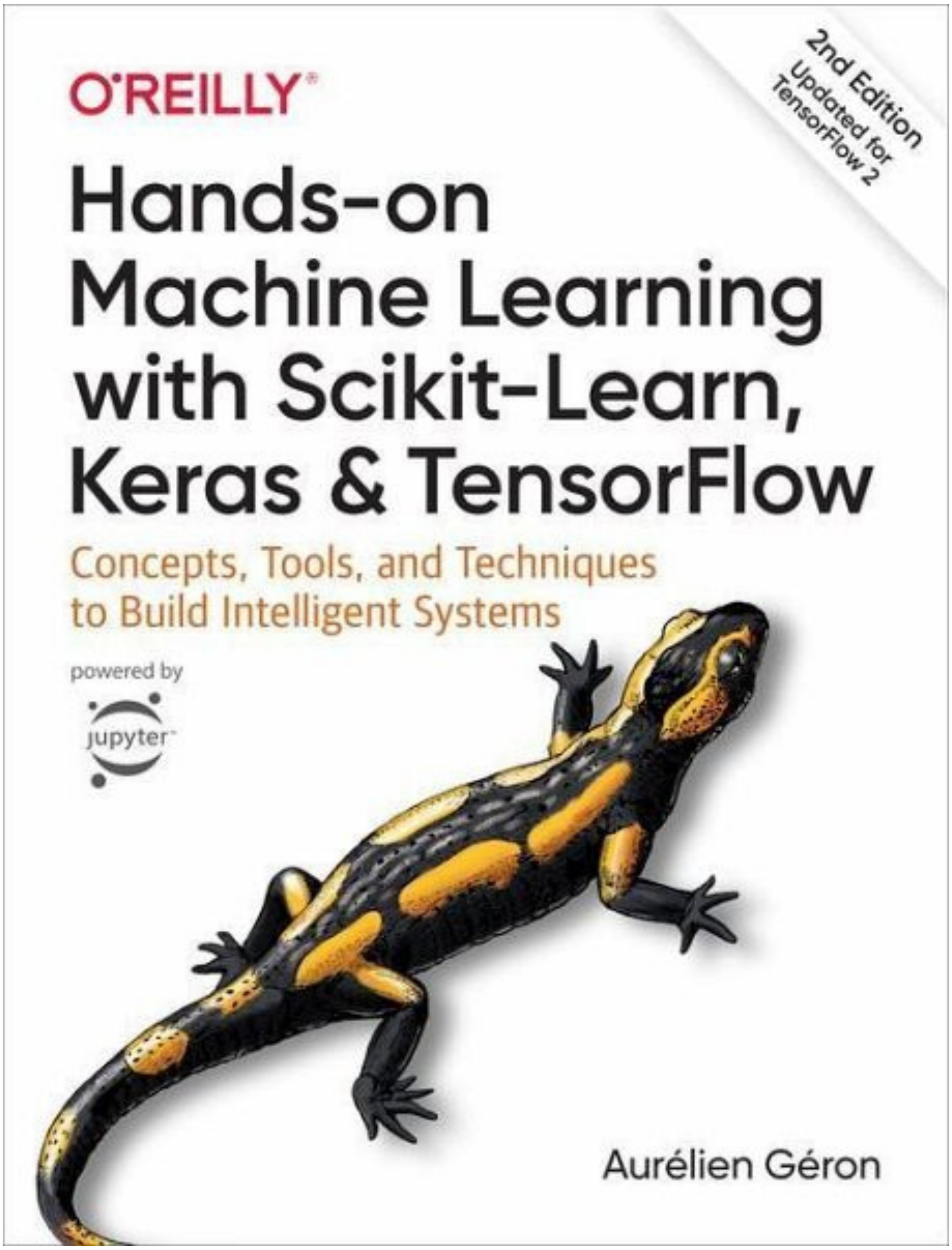
Book summary - First chapter

دوره اول
شهریور ۹۹

Machine learning landscape


O'REILLY
2nd Edition
Updated for TensorFlow 2
Hands-on Machine Learning with Scikit-Learn, Keras & TensorFlow
Concepts, Tools, and Techniques to Build Intelligent Systems
powered by
jupyter
Aurélien Géron

Hands-on machine learning with Scikit-learn

First chapter: Machine learning landscape

What is Machine learning?
Machine Learning is the science (and art) of programming computers so they can *learn from data*.

What is data mining?
Applying **ML techniques** to Dig into large amounts of data can help **discover patterns** that were not immediately apparent.

Machine Learning is great for:

- Problems for which existing solutions require a lot of fine-tuning or long lists of rules: one Machine Learning algorithm can often simplify code and perform better than the traditional approach.

- Complex problems for which using a traditional approach yields no good solution: the best Machine Learning techniques can perhaps find a solution.

- Fluctuating environments: a Machine Learning system can adapt to new data.

- Getting insights about complex problems and large amounts of data.

and not great for :

- **Deterministic problems :** Weather prediction, its possible with neural network but its computationally super expensive while a simple predictive algorithm can do enough good.
- **Lack of data or lack of good data:**
- **p hacking:** when researchers torture data to it confesses.
- **when interpretability is important**

Types of Machine learning system:

1. Whether or not they are trained with human supervision: Supervised, Unsupervised, semi-supervised and reinforcement learning

2. Online learning versus batch learning: While in online learning the system weights in the algorithm constantly keeps updating in batch learning it remains constant and the algorithm won't learn incrementally.
3. Instance based algorithms or model based: ?

quiz:
Is this sentence correct? 'We never use regression algorithms for classification problems'
False!
Note that some regression algorithms can be used for classification as well, and vice versa. For example, *Logistic Regression* is commonly used for classification, as it can output a value that corresponds to the probability of belonging to a given class (e.g., 20% chance of being spam).

Supervised algorithms:
- k-Nearest Neighbors
- Linear Regression
- Logistic Regression
- Support Vector Machines (SVMs)
- Decision Trees and Random Forests
- Neural networks[2]

Unsupervised algorithms:
- Clustering
  - K-Means
  - DBSCAN
  - Hierarchical Cluster Analysis (HCA)
- Anomaly detection and novelty detection
  - One-class SVM
  - Isolation Forest
- Visualization and dimensionality reduction
  - Principal Component Analysis (PCA)
  - Kernel PCA
  - Locally Linear Embedding (LLE)
  - t-Distributed Stochastic Neighbor Embedding (t-SNE)
- Association rule learning (dig into large amounts of data and find interesting relations between attributes)
  - Apriori

- Apriori
- Eclat

quiz:
Which option is not considered as a challenge for machine learning?

- Irrelevant features
- When the problem is too complex
- poor quality data
- Non-representative Training data

Overfitting happens when the model is too complex relative to the amount and noisiness of the training data. Here are possible solutions:

- Simplify the model by selecting one with fewer parameters (e.g., a linear model rather than a high-degree polynomial model), by reducing the number of attributes in the training data, or by constraining the model.
- Gather more training data.
- Reduce the noise in the training data (e.g., fix data errors and remove outliers).

Underfitting is opposite of overfitting means that the model is too simple to learn the underlying structure of the data

How to fix underfitting problem:

- Select a more powerful model, with more parameters.
- Feed better features to the learning algorithm (feature engineering).
- Reduce the constraints on the model (e.g., reduce the regularization hyperparameter).

Important:
The system will not perform well if your training set is too small, or if the data is not representative, is noisy, or is polluted with irrelevant features (garbage in, garbage out). Lastly, your model needs to be neither too simple (in which case it will underfit) nor too complex (in which case it will overfit).

quiz
what is generalisation error in a ML model?

- Error rate on training set
- Error rate on new cases

quiz
when the training error is low but generalisation error is high:

- Model is overfitting the training data
- Model is underfitting the training data
- Data is not representative

Don't iterate on the test dataset too much!
When we measure the generalisation error of our model with the test dataset for so many times, the model get trained just for the specific dataset. How to avoid that?
A common solution to this problem is called ***holdout validation***: you simply hold out part of the training set to evaluate several candidate models and select the best one. The new held-out set is called the *validation set* (or sometimes the *development set*, or *dev set*). More specifically, you train multiple models with various hyper parameters on the reduced training set (i.e., the full training set minus the validation set), and you select the model that performs best on the validation set. After this holdout validation process, you train the best model on the full training set (including the validation set), and this gives you the final model. Lastly, you evaluate this final model on the test set to get an estimate of the generalisation error.
What is the drawback? Training time is multiplied by the number of validation set.

Data Mismatch
When the model is performing poorly how to understand it is because of the overfitting the training set or having not representative training set?

One solution is to hold out some of the training set in yet another set that Andrew Ng calls the *train-dev set*. After the model is trained (on the training set, *not* on the train-dev set), you can evaluate it on the train-dev set. If it performs well, then the model is not overfitting the training set. If it performs poorly on the validation set, the problem must be coming from the data mismatch. Conversely, if the model performs poorly on the train-dev set, then it must have overfit the training set, so you should try to simplify or regularise the model, get more training data, and clean up the training data.

Whats the difference between validation set and train-dev?
In the former we train the model on reduced training set (training set - validation set) and then we choose the best model on full training set. afterwards we evaluate the final model on the test set. But regarding to train-dev, we generally use that when we have a great difference between the error rate test and training set! In this situation, we train the model

fully on the reduced training set (training set- train-dev) and tune hyper parameters fully with that, afterwards we evaluate the model on train-dev (like an additional test set) to see the limitation of the model is because of the non-representative dataset or overfitting the training set. Apparently if the model shows great results on train-dev means that the data is not representative and conversely if we get poor result on train-dev it shows the model is overfitted!