Disclaimer

This document is a summary that follows the content of *Hands-on machine learning with Scikit-Learn, Keras and TensorFlow 2nd ed.*.This work is published as CC BY-NC-SA.

I do not guarantee correctness or completeness, nor is this document endorsed by the writers. Feel free to point out any erratas. For the full LaTeX source code, consider .

The Machine Learning Landscape

What is Machine Learning?

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

Here is a slightly more general definition: [Machine Learning is the] field of study that gives computers the ability to learn without being explicitly programmed.
—Arthur Samuel, 1959

And a more engineering-oriented one: A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.
—Tom Mitchell, 1997

Why use Machine Learning?

To summarize, 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.

Types of Machine Learning Systems

There are so many different types of Machine Learning systems that it is useful to classify them in broad categories, based on the following criteria:

• Whether or not they are trained with human supervision (supervised, unsuper- vised, semisupervised, and Reinforcement Learning)

• Whether or not they can learn incrementally on the fly (online versus batch learning)

• Whether they work by simply comparing new data points to known data points, or instead by detecting patterns in the training data and building a predictive model, much like scientists do (instance-based versus model-based learning)

Supervised/Unsupervised Learning

In ***supervised learning***, the training set you feed to the algorithm includes the desired solutions, called labels

Here are some of the most important supervised learning algorithms (covered in this book):

• k-Nearest Neighbors
• Linear Regression
• Logistic Regression
• Support Vector Machines (SVMs)
• Decision Trees and Random Forests
• Neural networks

In ***unsupervised learning*** , the training data is unlabeled. The system tries to learn without a teacher. Here are some of the most important unsupervised learning 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
— Apriori
— Eclat

Since labeling data is usually time-consuming and costly, you will often have plenty of unlabeled instances, and few labeled instances. Some algorithms can deal with data that's partially labeled. This is called ***semisupervised learning***

Reinforcement Learning is a very different beast. The learning system, called an agent in this context, can observe the environment, select and perform actions, and get rewards in return or penalties.

It must then learn by itself what is the best strategy, called a policy, to get the most reward over time. A policy defines what action the agent should choose when it is in a given situation.

Batch and Online Learning

In ***batch learning***, the system is incapable of learning incrementally: it must be trained using all the available data. This will generally take a lot of time and computing resources, so it is typically done offline. First the system is trained, and then it is launched into production and runs without learning anymore; it just applies what it has learned. This is called *offline learning*.

In ***online learning***, you train the system incrementally by feeding it data instances sequentially, either individually or in small groups called mini-batches. Each learning step is fast and cheap, so the system can learn about new data on the fly, as it arrives. One important parameter of online learning systems is how fast they should adapt to changing data: this is called the *learning rate*.

Instance-Based Versus Model-Based Learning

One more way to categorize Machine Learning systems is by how they *generalize*. ***instance-based learning***: the system learns the examples by heart, then generalizes to new cases by using a similarity measure to compare them to the learned examples (or a subset of them).

Another way to generalize from a set of examples is to build a model of these exam- ples and then use that model to make predictions. This is called ***model-based learning***

Overfitting the training Data

Overfitting: it means that the model performs well on the training data, but it does not generalize well.

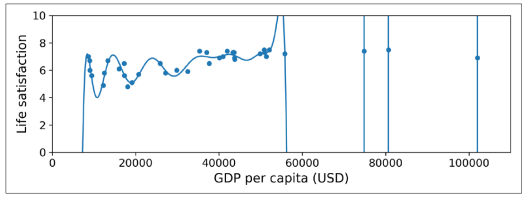

Figure 1-22. Overfitting the training data

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 the training Data

underfitting is the opposite of overfitting: it occurs when your model is too simple to learn the underlying structure of the data. Here are the main options for fixing this 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 hyperpara- meter).

Testing and Validating

Split your data into two sets: the training set and the test set. As these names imply, you train your model using the training set, and you test it using the test set. The error rate on new cases is called the *generalization error*.

If the training error is low (i.e., your model makes few mistakes on the training set) but the generalization error is high, it means that your model is overfitting the train- ing data.

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 hyperparameters 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 vali- dation 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 generalization error.

cross-validation: using many small validation sets. Each model is evaluated once per validation set after it is trained on the rest of the data. By averaging out all the evaluations of a model, you get a much more accurate measure of its perfor- mance. There is a drawback, however: the training time is multiplied by the number of validation sets.