

# Hands-On Machine Learning with Scikit-Learn & TensorFlow

CONCEPTS, TOOLS, AND TECHNIQUES  
TO BUILD INTELLIGENT SYSTEMS



powered by



**easy** ●●●  
**computing**

Aurélien Géron

# Hands-On Machine Learning with Scikit-Learn and TensorFlow

Through a series of recent breakthroughs, deep learning has boosted the entire field of machine learning. Now, even programmers who know close to nothing about this technology can use simple, efficient tools to implement programs capable of learning from data. This practical book shows you how.

By using concrete examples, minimal theory, and two production-ready Python frameworks—Scikit-Learn and TensorFlow—author Aurélien Géron helps you gain an intuitive understanding of the concepts and tools for building intelligent systems. You'll learn a range of techniques, starting with simple linear regression and progressing to deep neural networks. With exercises in each chapter to help you apply what you've learned, all you need is programming experience to get started.

“This book is a great introduction to the theory and practice of solving problems with neural networks. It covers the key points you'll need to build effective applications, along with enough background to understand new research as it emerges. I recommend this book to anyone interested in learning about practical ML.”

—Pete Warden  
Mobile Lead for TensorFlow

- Explore the machine learning landscape, particularly neural nets
- Use Scikit-Learn to track an example machine learning project end-to-end
- Explore several training models, including support vector machines, decision trees, random forests, and ensemble methods
- Use the TensorFlow library to build and train neural nets
- Dive into neural net architectures, including convolutional nets, recurrent nets, and deep reinforcement learning
- Learn techniques for training and scaling deep neural nets
- Apply practical code examples without acquiring excessive machine learning theory or algorithm details

**Aurélien Géron** is a machine learning consultant. A former Googler, he led the YouTube video classification team from 2013 to 2016. He was also a founder and CTO of Wifirst from 2002 to 2012, a leading Wireless ISP in France, and a founder and CTO of Polyconseil in 2001, the firm that now manages the electric car sharing service Autolib'.

DATA | DATA SCIENCE | DATA ANALYTICS | MACHINE LEARNING |  
DEEP LEARNING | PYTHON MACHINE LEARNING

US \$49.99

CAN \$65.99

ISBN: 978-1-491-96229-9

easy  
computing



Twitter: @oreillymedia  
facebook.com/oreilly

---

# Hands-On Machine Learning with Scikit-Learn and TensorFlow

*Concepts, Tools, and Techniques to  
Build Intelligent Systems*

*Aurélien Géron*

easy   
computing

Beijing • Boston • Farnham • Sebastopol • Tokyo

**O'REILLY®**

# Hands-On Machine Learning with Scikit-Learn and TensorFlow

by Aurélien Géron

Copyright © 2017 Aurélien Géron. All rights reserved.

Printed in the United States of America.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (<http://oreilly.com/safari>). For more information, contact our corporate/institutional sales department: 800-998-9938 or [corporate@oreilly.com](mailto:corporate@oreilly.com).

**Editor:** Nicole Tache

**Production Editor:** Nicholas Adams

**Copyeditor:** Rachel Monaghan

**Proofreader:** Charles Roumeliotis

**Indexer:** Wendy Catalano

**Interior Designer:** David Futato

**Cover Designer:** Randy Comer

**Illustrator:** Rebecca Demarest

March 2017: First Edition

## Revision History for the First Edition

2017-03-10: First Release

2017-06-09: Second Release

2017-08-18: Third Release

2017-11-03: Fourth Release

2018-01-19: Fifth Release

See <http://oreilly.com/catalog/errata.csp?isbn=9781491962299> for release details.

The O'Reilly logo is a registered trademark of O'Reilly Media, Inc. *Hands-On Machine Learning with Scikit-Learn and TensorFlow*, the cover image, and related trade dress are trademarks of O'Reilly Media, Inc.

While the publisher and the author have used good faith efforts to ensure that the information and instructions contained in this work are accurate, the publisher and the author disclaim all responsibility for errors or omissions, including without limitation responsibility for damages resulting from the use of or reliance on this work. Use of the information and instructions contained in this work is at your own risk. If any code samples or other technology this work contains or describes is subject to open source licenses or the intellectual property rights of others, it is your responsibility to ensure that your use thereof complies with such licenses and/or rights.

978-1-491-96229-9

[M]

easy   
computing

---

# Table of Contents

Preface.....	xiii
--------------	------

---

## Part I. The Fundamentals of Machine Learning

<b>1. The Machine Learning Landscape.....</b>	<b>3</b>
What Is Machine Learning?	4
Why Use Machine Learning?	4
Types of Machine Learning Systems	7
Supervised/Unsupervised Learning	8
Batch and Online Learning	14
Instance-Based Versus Model-Based Learning	17
Main Challenges of Machine Learning	22
Insufficient Quantity of Training Data	22
Nonrepresentative Training Data	24
Poor-Quality Data	25
Irrelevant Features	25
Overfitting the Training Data	26
Underfitting the Training Data	28
Stepping Back	28
Testing and Validating	29
Exercises	31
<b>2. End-to-End Machine Learning Project.....</b>	<b>33</b>
Working with Real Data	33
Look at the Big Picture	35
Frame the Problem	35
Select a Performance Measure	37

Check the Assumptions	40
Get the Data	40
Create the Workspace	40
Download the Data	43
Take a Quick Look at the Data Structure	45
Create a Test Set	49
Discover and Visualize the Data to Gain Insights	53
Visualizing Geographical Data	53
Looking for Correlations	56
Experimenting with Attribute Combinations	59
Prepare the Data for Machine Learning Algorithms	60
Data Cleaning	61
Handling Text and Categorical Attributes	63
Custom Transformers	65
Feature Scaling	66
Transformation Pipelines	67
Select and Train a Model	69
Training and Evaluating on the Training Set	69
Better Evaluation Using Cross-Validation	71
Fine-Tune Your Model	73
Grid Search	73
Randomized Search	75
Ensemble Methods	76
Analyze the Best Models and Their Errors	76
Evaluate Your System on the Test Set	77
Launch, Monitor, and Maintain Your System	78
Try It Out!	78
Exercises	79
 <b>3. Classification.....</b>	 <b>81</b>
MNIST	81
Training a Binary Classifier	84
Performance Measures	84
Measuring Accuracy Using Cross-Validation	85
Confusion Matrix	86
Precision and Recall	88
Precision/Recall Tradeoff	89
The ROC Curve	93
Multiclass Classification	95
Error Analysis	98
Multilabel Classification	102
Multioutput Classification	103

Exercises	104
<b>4. Training Models.....</b>	<b>107</b>
Linear Regression	108
The Normal Equation	110
Computational Complexity	112
Gradient Descent	113
Batch Gradient Descent	116
Stochastic Gradient Descent	119
Mini-batch Gradient Descent	121
Polynomial Regression	123
Learning Curves	125
Regularized Linear Models	129
Ridge Regression	129
Lasso Regression	132
Elastic Net	134
Early Stopping	135
Logistic Regression	136
Estimating Probabilities	136
Training and Cost Function	137
Decision Boundaries	139
Softmax Regression	141
Exercises	145
<b>5. Support Vector Machines.....</b>	<b>147</b>
Linear SVM Classification	147
Soft Margin Classification	148
Nonlinear SVM Classification	151
Polynomial Kernel	152
Adding Similarity Features	153
Gaussian RBF Kernel	154
Computational Complexity	156
SVM Regression	156
Under the Hood	158
Decision Function and Predictions	158
Training Objective	159
Quadratic Programming	161
The Dual Problem	162
Kernelized SVM	163
Online SVMs	166
Exercises	167

<b>6. Decision Trees.....</b>	<b>169</b>
Training and Visualizing a Decision Tree	169
Making Predictions	171
Estimating Class Probabilities	173
The CART Training Algorithm	173
Computational Complexity	174
Gini Impurity or Entropy?	174
Regularization Hyperparameters	175
Regression	177
Instability	179
Exercises	180
<b>7. Ensemble Learning and Random Forests.....</b>	<b>183</b>
Voting Classifiers	183
Bagging and Pasting	187
Bagging and Pasting in Scikit-Learn	188
Out-of-Bag Evaluation	189
Random Patches and Random Subspaces	190
Random Forests	191
Extra-Trees	192
Feature Importance	192
Boosting	193
AdaBoost	194
Gradient Boosting	197
Stacking	202
Exercises	204
<b>8. Dimensionality Reduction.....</b>	<b>207</b>
The Curse of Dimensionality	208
Main Approaches for Dimensionality Reduction	209
Projection	209
Manifold Learning	212
PCA	213
Preserving the Variance	213
Principal Components	214
Projecting Down to d Dimensions	215
Using Scikit-Learn	216
Explained Variance Ratio	216
Choosing the Right Number of Dimensions	217
PCA for Compression	218
Incremental PCA	219
Randomized PCA	220



Kernel PCA	220
Selecting a Kernel and Tuning Hyperparameters	221
LLE	223
Other Dimensionality Reduction Techniques	225
Exercises	226

---

## Part II. Neural Networks and Deep Learning

<b>9. Up and Running with TensorFlow.....</b>	<b>231</b>
Installation	234
Creating Your First Graph and Running It in a Session	234
Managing Graphs	236
Lifecycle of a Node Value	237
Linear Regression with TensorFlow	237
Implementing Gradient Descent	239
Manually Computing the Gradients	239
Using autodiff	240
Using an Optimizer	241
Feeding Data to the Training Algorithm	241
Saving and Restoring Models	243
Visualizing the Graph and Training Curves Using TensorBoard	244
Name Scopes	247
Modularity	248
Sharing Variables	250
Exercises	253
<b>10. Introduction to Artificial Neural Networks.....</b>	<b>255</b>
From Biological to Artificial Neurons	256
Biological Neurons	257
Logical Computations with Neurons	258
The Perceptron	259
Multi-Layer Perceptron and Backpropagation	263
Training an MLP with TensorFlow's High-Level API	266
Training a DNN Using Plain TensorFlow	267
Construction Phase	267
Execution Phase	271
Using the Neural Network	272
Fine-Tuning Neural Network Hyperparameters	272
Number of Hidden Layers	273
Number of Neurons per Hidden Layer	274
Activation Functions	274



<b>11. Training Deep Neural Nets.....</b>	<b>277</b>
Vanishing/Exploding Gradients Problems	277
Xavier and He Initialization	279
Nonsaturating Activation Functions	281
Batch Normalization	284
Gradient Clipping	288
Reusing Pretrained Layers	289
Reusing a TensorFlow Model	289
Reusing Models from Other Frameworks	291
Freezing the Lower Layers	292
Caching the Frozen Layers	293
Tweaking, Dropping, or Replacing the Upper Layers	294
Model Zoos	294
Unsupervised Pretraining	295
Pretraining on an Auxiliary Task	296
Faster Optimizers	297
Momentum Optimization	297
Nesterov Accelerated Gradient	299
AdaGrad	300
RMSProp	302
Adam Optimization	302
Learning Rate Scheduling	305
Avoiding Overfitting Through Regularization	307
Early Stopping	307
$\ell_1$ and $\ell_2$ Regularization	307
Dropout	309
Max-Norm Regularization	311
Data Augmentation	313
Practical Guidelines	314
Exercises	315
<b>12. Distributing TensorFlow Across Devices and Servers.....</b>	<b>317</b>
Multiple Devices on a Single Machine	318
Installation	318
Managing the GPU RAM	321
Placing Operations on Devices	322
Parallel Execution	325
Control Dependencies	327
Multiple Devices Across Multiple Servers	328
Opening a Session	330



The Master and Worker Services	330
Pinning Operations Across Tasks	331
Sharding Variables Across Multiple Parameter Servers	331
Sharing State Across Sessions Using Resource Containers	332
Asynchronous Communication Using TensorFlow Queues	334
Loading Data Directly from the Graph	339
Parallelizing Neural Networks on a TensorFlow Cluster	346
One Neural Network per Device	346
In-Graph Versus Between-Graph Replication	347
Model Parallelism	350
Data Parallelism	352
Exercises	357
<b>13. Convolutional Neural Networks.....</b>	<b>359</b>
The Architecture of the Visual Cortex	360
Convolutional Layer	361
Filters	363
Stacking Multiple Feature Maps	364
TensorFlow Implementation	366
Memory Requirements	368
Pooling Layer	369
CNN Architectures	371
LeNet-5	372
AlexNet	373
GoogLeNet	375
ResNet	378
Exercises	382
<b>14. Recurrent Neural Networks.....</b>	<b>385</b>
Recurrent Neurons	386
Memory Cells	388
Input and Output Sequences	389
Basic RNNs in TensorFlow	390
Static Unrolling Through Time	391
Dynamic Unrolling Through Time	393
Handling Variable Length Input Sequences	394
Handling Variable-Length Output Sequences	395
Training RNNs	395
Training a Sequence Classifier	396
Training to Predict Time Series	398
Creative RNN	402
Deep RNNs	403

Distributing a Deep RNN Across Multiple GPUs	404
Applying Dropout	405
The Difficulty of Training over Many Time Steps	406
LSTM Cell	407
Peephole Connections	410
GRU Cell	410
Natural Language Processing	412
Word Embeddings	412
An Encoder–Decoder Network for Machine Translation	414
Exercises	417
<b>15. Autoencoders.....</b>	<b>419</b>
Efficient Data Representations	420
Performing PCA with an Undercomplete Linear Autoencoder	421
Stacked Autoencoders	423
TensorFlow Implementation	424
Tying Weights	425
Training One Autoencoder at a Time	426
Visualizing the Reconstructions	429
Visualizing Features	429
Unsupervised Pretraining Using Stacked Autoencoders	430
Denoising Autoencoders	432
TensorFlow Implementation	433
Sparse Autoencoders	434
TensorFlow Implementation	436
Variational Autoencoders	437
Generating Digits	440
Other Autoencoders	441
Exercises	442
<b>16. Reinforcement Learning.....</b>	<b>445</b>
Learning to Optimize Rewards	446
Policy Search	448
Introduction to OpenAI Gym	449
Neural Network Policies	453
Evaluating Actions: The Credit Assignment Problem	455
Policy Gradients	456
Markov Decision Processes	461
Temporal Difference Learning and Q-Learning	465
Exploration Policies	467
Approximate Q-Learning and Deep Q-Learning	468
Learning to Play Ms. Pac-Man Using the DQN Algorithm	469

Exercises	477
Thank You!	478
<b>A. Exercise Solutions.....</b>	<b>479</b>
<b>B. Machine Learning Project Checklist.....</b>	<b>505</b>
<b>C. SVM Dual Problem.....</b>	<b>511</b>
<b>D. Autodiff.....</b>	<b>515</b>
<b>E. Other Popular ANN Architectures.....</b>	<b>523</b>
<b>Index.....</b>	<b>533</b>

**easy ●●●**  
**computing**

---

# Preface

## The Machine Learning Tsunami

In 2006, Geoffrey Hinton et al. published a paper<sup>1</sup> showing how to train a deep neural network capable of recognizing handwritten digits with state-of-the-art precision (>98%). They branded this technique “Deep Learning.” Training a deep neural net was widely considered impossible at the time,<sup>2</sup> and most researchers had abandoned the idea since the 1990s. This paper revived the interest of the scientific community and before long many new papers demonstrated that Deep Learning was not only possible, but capable of mind-blowing achievements that no other Machine Learning (ML) technique could hope to match (with the help of tremendous computing power and great amounts of data). This enthusiasm soon extended to many other areas of Machine Learning.

Fast-forward 10 years and Machine Learning has conquered the industry: it is now at the heart of much of the magic in today’s high-tech products, ranking your web search results, powering your smartphone’s speech recognition, and recommending videos, beating the world champion at the game of Go. Before you know it, it will be driving your car.

## Machine Learning in Your Projects

So naturally you are excited about Machine Learning and you would love to join the party!

Perhaps you would like to give your homemade robot a brain of its own? Make it recognize faces? Or learn to walk around?

---

1 Available on Hinton’s home page at <http://www.cs.toronto.edu/~hinton/>.

2 Despite the fact that Yann LeCun’s deep convolutional neural networks had worked well for image recognition since the 1990s, although they were not as general purpose.

Or maybe your company has tons of data (user logs, financial data, production data, machine sensor data, hotline stats, HR reports, etc.), and more than likely you could unearth some hidden gems if you just knew where to look; for example:

- Segment customers and find the best marketing strategy for each group
- Recommend products for each client based on what similar clients bought
- Detect which transactions are likely to be fraudulent
- Predict next year's revenue
- **And more**

Whatever the reason, you have decided to learn Machine Learning and implement it in your projects. Great idea!

## Objective and Approach

This book assumes that you know close to nothing about Machine Learning. Its goal is to give you the concepts, the intuitions, and the tools you need to actually implement programs capable of *learning from data*.

We will cover a large number of techniques, from the simplest and most commonly used (such as linear regression) to some of the Deep Learning techniques that regularly win competitions.

Rather than implementing our own toy versions of each algorithm, we will be using actual production-ready Python frameworks:

- **Scikit-Learn** is very easy to use, yet it implements many Machine Learning algorithms efficiently, so it makes for a great entry point to learn Machine Learning.
- **TensorFlow** is a more complex library for distributed numerical computation using data flow graphs. It makes it possible to train and run very large neural networks efficiently by distributing the computations across potentially thousands of multi-GPU servers. TensorFlow was created at Google and supports many of their large-scale Machine Learning applications. It was open-sourced in November 2015.

The book favors a hands-on approach, growing an intuitive understanding of Machine Learning through concrete working examples and just a little bit of theory. While you can read this book without picking up your laptop, we highly recommend you experiment with the code examples available online as Jupyter notebooks at <https://github.com/ageron/handson-ml>.





# Prerequisites

This book assumes that you have some Python programming experience and that you are familiar with Python's main scientific libraries, in particular **NumPy**, **Pandas**, and **Matplotlib**.

Also, if you care about what's under the hood you should have a reasonable understanding of college-level math as well (calculus, linear algebra, probabilities, and statistics).

If you don't know Python yet, <http://learnpython.org/> is a great place to start. The official tutorial on [python.org](http://python.org) is also quite good.

If you have never used Jupyter, **Chapter 2** will guide you through installation and the basics: it is a great tool to have in your toolbox.

If you are not familiar with Python's scientific libraries, the provided Jupyter notebooks include a few tutorials. There is also a quick math tutorial for linear algebra.

# Roadmap

This book is organized in two parts. **Part I, *The Fundamentals of Machine Learning***, covers the following topics:

- What is Machine Learning? What problems does it try to solve? What are the main categories and fundamental concepts of Machine Learning systems?
- The main steps in a typical Machine Learning project.
- Learning by fitting a model to data.
- Optimizing a cost function.
- Handling, cleaning, and preparing data.
- Selecting and engineering features.
- Selecting a model and tuning hyperparameters using cross-validation.
- The main challenges of Machine Learning, in particular underfitting and overfitting (the bias/variance tradeoff).
- Reducing the dimensionality of the training data to fight the curse of dimensionality.
- The most common learning algorithms: Linear and Polynomial Regression, Logistic Regression, k-Nearest Neighbors, Support Vector Machines, Decision Trees, Random Forests, and Ensemble methods.



Part II, *Neural Networks and Deep Learning*, covers the following topics:

- What are neural nets? What are they good for?
- Building and training neural nets using TensorFlow.
- The most important neural net architectures: feedforward neural nets, convolutional nets, recurrent nets, long short-term memory (LSTM) nets, and autoencoders.
- Techniques for training deep neural nets.
- Scaling neural networks for huge datasets.
- Reinforcement learning.

The first part is based mostly on Scikit-Learn while the second part uses TensorFlow.



Don't jump into deep waters too hastily: while Deep Learning is no doubt one of the most exciting areas in Machine Learning, you should master the fundamentals first. Moreover, most problems can be solved quite well using simpler techniques such as Random Forests and Ensemble methods (discussed in [Part I](#)). Deep Learning is best suited for complex problems such as image recognition, speech recognition, or natural language processing, provided you have enough data, computing power, and patience.

## Other Resources

Many resources are available to learn about Machine Learning. Andrew Ng's [ML course on Coursera](#) and Geoffrey Hinton's [course on neural networks and Deep Learning](#) are amazing, although they both require a significant time investment (think months).

There are also many interesting websites about Machine Learning, including of course Scikit-Learn's exceptional [User Guide](#). You may also enjoy [Dataquest](#), which provides very nice interactive tutorials, and ML blogs such as those listed on [Quora](#). Finally, the [Deep Learning website](#) has a good list of resources to learn more.

Of course there are also many other introductory books about Machine Learning, in particular:

- Joel Grus, *Data Science from Scratch* (O'Reilly). This book presents the fundamentals of Machine Learning, and implements some of the main algorithms in pure Python (from scratch, as the name suggests).
- Stephen Marsland, *Machine Learning: An Algorithmic Perspective* (Chapman and Hall). This book is a great introduction to Machine Learning, covering a wide

easy computing

range of topics in depth, with code examples in Python (also from scratch, but using NumPy).

- Sebastian Raschka, *Python Machine Learning* (Packt Publishing). Also a great introduction to Machine Learning, this book leverages Python open source libraries (Pylearn 2 and Theano).
- Yaser S. Abu-Mostafa, Malik Magdon-Ismail, and Hsuan-Tien Lin, *Learning from Data* (AMLLBook). A rather theoretical approach to ML, this book provides deep insights, in particular on the bias/variance tradeoff (see [Chapter 4](#)).
- Stuart Russell and Peter Norvig, *Artificial Intelligence: A Modern Approach, 3rd Edition* (Pearson). This is a great (and huge) book covering an incredible amount of topics, including Machine Learning. It helps put ML into perspective.

Finally, a great way to learn is to join ML competition websites such as [Kaggle.com](#) this will allow you to practice your skills on real-world problems, with help and insights from some of the best ML professionals out there.

## Conventions Used in This Book

The following typographical conventions are used in this book:

### *Italic*

Indicates new terms, URLs, email addresses, filenames, and file extensions.

### Constant width

Used for program listings, as well as within paragraphs to refer to program elements such as variable or function names, databases, data types, environment variables, statements and keywords.

### Constant width bold

Shows commands or other text that should be typed literally by the user.

### Constant width italic

Shows text that should be replaced with user-supplied values or by values determined by context.



This element signifies a tip or suggestion.



This element signifies a general note.



This element indicates a warning or caution.

## Using Code Examples

Supplemental material (code examples, exercises, etc.) is available for download at <https://github.com/ageron/handson-ml>.

This book is here to help you get your job done. In general, if example code is offered with this book, you may use it in your programs and documentation. You do not need to contact us for permission unless you’re reproducing a significant portion of the code. For example, writing a program that uses several chunks of code from this book does not require permission. Selling or distributing a CD-ROM of examples from O’Reilly books does require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product’s documentation does require permission.

We appreciate, but do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: “*Hands-On Machine Learning with Scikit-Learn and TensorFlow* by Aurélien Géron (O’Reilly). Copyright 2017 Aurélien Géron, 978-1-491-96229-9.”

If you feel your use of code examples falls outside fair use or the permission given above, feel free to contact us at [permissions@oreilly.com](mailto:permissions@oreilly.com).

## O’Reilly Safari



**Safari**<sup>®</sup>

*Safari* (formerly Safari Books Online) is a membership-based training and reference platform for enterprise, government, educators, and individuals.

Members have access to thousands of books, training videos, Learning Paths, interactive tutorials, and curated playlists from over 250 publishers, including O’Reilly Media, Harvard Business Press, Morgan Kaufmann Professional, Addison-Wesley Professional, Microsoft Press, Sams, Que, Peachpit Press, Adobe, Focal Press, Cisco Press,

easy  
computing

John Wiley & Sons, Syngress, Morgan Kaufmann, IBM Redbooks, Packt, Adobe Press, FT Press, Apress, Manning, New Riders, McGraw-Hill, Jones & Bartlett, and Course Technology, among others.

For more information, please visit <http://oreilly.com/safari>.

## How to Contact Us

Please address comments and questions concerning this book to the publisher:

O'Reilly Media, Inc.  
1005 Gravenstein Highway North  
Sebastopol, CA 95472  
800-998-9938 (in the United States or Canada)  
707-829-0515 (international or local)  
707-829-0104 (fax)

We have a web page for this book, where we list errata, examples, and any additional information. You can access this page at <http://bit.ly/hands-on-machine-learning-with-scikit-learn-and-tensorflow>.

To comment or ask technical questions about this book, send email to [bookquestions@oreilly.com](mailto:bookquestions@oreilly.com).

For more information about our books, courses, conferences, and news, see our website at <http://www.oreilly.com>.

Find us on Facebook: <http://facebook.com/oreilly>

Follow us on Twitter: <http://twitter.com/oreillymedia>

Watch us on YouTube: <http://www.youtube.com/oreillymedia>

## Acknowledgments

I would like to thank my Google colleagues, in particular the YouTube video classification team, for teaching me so much about Machine Learning. I could never have started this project without them. Special thanks to my personal ML gurus: Clément Courbet, Julien Dubois, Mathias Kende, Daniel Kitachewsky, James Pack, Alexander Pak, Anosh Raj, Vitor Sessak, Wiktor Tomczak, Ingrid von Glehn, Rich Washington, and everyone at YouTube Paris.

I am incredibly grateful to all the amazing people who took time out of their busy lives to review my book in so much detail. Thanks to Pete Warden for answering all my TensorFlow questions, reviewing [Part II](#), providing many interesting insights, and of course for being part of the core TensorFlow team. You should definitely check out



[his blog](#)! Many thanks to Lukas Biewald for his very thorough review of [Part II](#): he left no stone unturned, tested all the code (and caught a few errors), made many great suggestions, and his enthusiasm was contagious. You should check out [his blog](#) and his [cool robots](#)! Thanks to Justin Francis, who also reviewed [Part II](#) very thoroughly, catching errors and providing great insights, in particular in [Chapter 16](#). Check out [his posts](#) on TensorFlow!

Huge thanks as well to David Andrzejewski, who reviewed [Part I](#) and provided incredibly useful feedback, identifying unclear sections and suggesting how to improve them. Check out [his website](#)! Thanks to Grégoire Mesnil, who reviewed [Part II](#) and contributed very interesting practical advice on training neural networks. Thanks as well to Eddy Hung, Salim Sémaoune, Karim Matrah, Ingrid von Glehn, Iain Smears, and Vincent Guilbeau for reviewing [Part I](#) and making many useful suggestions. And I also wish to thank my father-in-law, Michel Tessier, former mathematics teacher and now a great translator of Anton Chekhov, for helping me iron out some of the mathematics and notations in this book and reviewing the linear algebra Jupyter notebook.

And of course, a gigantic “thank you” to my dear brother Sylvain, who reviewed every single chapter, tested every line of code, provided feedback on virtually every section, and encouraged me from the first line to the last. Love you, bro!

Many thanks as well to O’Reilly’s fantastic staff, in particular Nicole Tache, who gave me insightful feedback, always cheerful, encouraging, and helpful. Thanks as well to Marie Beaugureau, Ben Lorica, Mike Loukides, and Laurel Ruma for believing in this project and helping me define its scope. Thanks to Matt Hacker and all of the Atlas team for answering all my technical questions regarding formatting, asciidoc, and LaTeX, and thanks to Rachel Monaghan, Nick Adams, and all of the production team for their final review and their hundreds of corrections.

Last but not least, I am infinitely grateful to my beloved wife, Emmanuelle, and to our three wonderful kids, Alexandre, Rémi, and Gabrielle, for encouraging me to work hard on this book, asking many questions (who said you can’t teach neural networks to a seven-year-old?), and even bringing me cookies and coffee. What more can one dream of?



PART I

---

# The Fundamentals of Machine Learning

**easy ●●●**  
**computing**



---

# The Machine Learning Landscape

When most people hear “Machine Learning,” they picture a robot: a dependable butler or a deadly Terminator depending on who you ask. But Machine Learning is not just a futuristic fantasy, it’s already here. In fact, it has been around for decades in some specialized applications, such as *Optical Character Recognition* (OCR). But the first ML application that really became mainstream, improving the lives of hundreds of millions of people, took over the world back in the 1990s: it was the *spam filter*. Not exactly a self-aware Skynet, but it does technically qualify as Machine Learning (it has actually learned so well that you seldom need to flag an email as spam anymore). It was followed by hundreds of ML applications that now quietly power hundreds of products and features that you use regularly, from better recommendations to voice search.

Where does Machine Learning start and where does it end? What exactly does it mean for a machine to *learn* something? If I download a copy of Wikipedia, has my computer really “learned” something? Is it suddenly smarter? In this chapter we will start by clarifying what Machine Learning is and why you may want to use it.

Then, before we set out to explore the Machine Learning continent, we will take a look at the map and learn about the main regions and the most notable landmarks: supervised versus unsupervised learning, online versus batch learning, instance-based versus model-based learning. Then we will look at the workflow of a typical ML project, discuss the main challenges you may face, and cover how to evaluate and fine-tune a Machine Learning system.

This chapter introduces a lot of fundamental concepts (and jargon) that every data scientist should know by heart. It will be a high-level overview (the only chapter without much code), all rather simple, but you should make sure everything is crystal-clear to you before continuing to the rest of the book. So grab a coffee and let’s get started!

easy ●●●  
computing



If you already know all the Machine Learning basics, you may want to skip directly to **Chapter 2**. If you are not sure, try to answer all the questions listed at the end of the chapter before moving on.

## 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

For example, your spam filter is a Machine Learning program that can learn to flag spam given examples of spam emails (e.g., flagged by users) and examples of regular (nospam, also called “ham”) emails. The examples that the system uses to learn are called the *training set*. Each training example is called a *training instance* (or *sample*). In this case, the task  $T$  is to flag spam for new emails, the experience  $E$  is the *training data*, and the performance measure  $P$  needs to be defined; for example, you can use the ratio of correctly classified emails. This particular performance measure is called *accuracy* and it is often used in classification tasks.

If you just download a copy of Wikipedia, your computer has a lot more data, but it is not suddenly better at any task. Thus, it is not Machine Learning.

## Why Use Machine Learning?

Consider how you would write a spam filter using traditional programming techniques (**Figure 1-1**):

1. First you would look at what spam typically looks like. You might notice that some words or phrases (such as “4U,” “credit card,” “free,” and “amazing”) tend to come up a lot in the subject. Perhaps you would also notice a few other patterns in the sender’s name, the email’s body, and so on.

easy   
computing

2. You would write a detection algorithm for each of the patterns that you noticed, and your program would flag emails as spam if a number of these patterns are detected.
3. You would test your program, and repeat steps 1 and 2 until it is good enough.

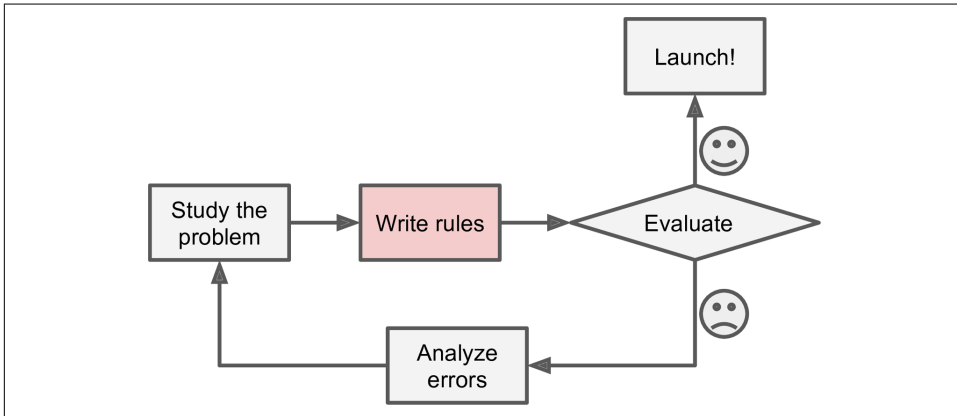


Figure 1-1. The traditional approach

Since the problem is not trivial, your program will likely become a long list of complex rules—pretty hard to maintain.

In contrast, a spam filter based on Machine Learning techniques automatically learns which words and phrases are good predictors of spam by detecting unusually frequent patterns of words in the spam examples compared to the ham examples (Figure 1-2). The program is much shorter, easier to maintain, and most likely more accurate.

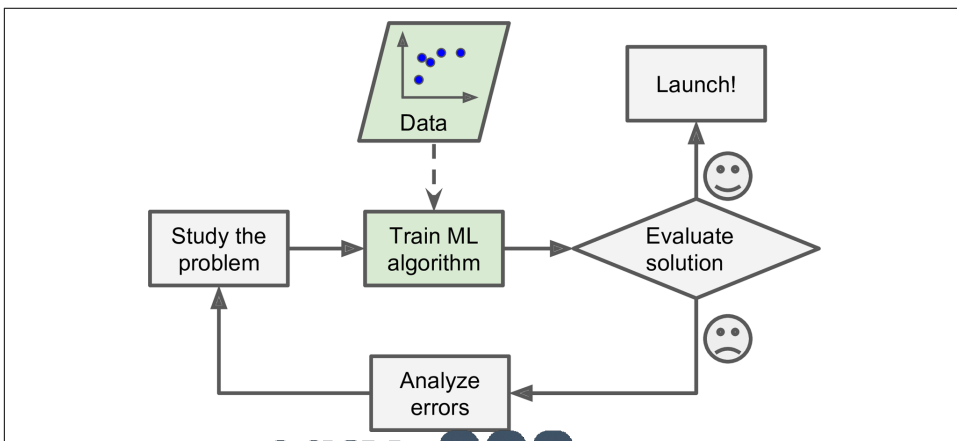


Figure 1-2. Machine Learning approach

Moreover, if spammers notice that all their emails containing “4U” are blocked, they might start writing “For U” instead. A spam filter using traditional programming techniques would need to be updated to flag “For U” emails. If spammers keep working around your spam filter, you will need to keep writing new rules forever.

In contrast, a spam filter based on Machine Learning techniques automatically notices that “For U” has become unusually frequent in spam flagged by users, and it starts flagging them without your intervention (Figure 1-3).

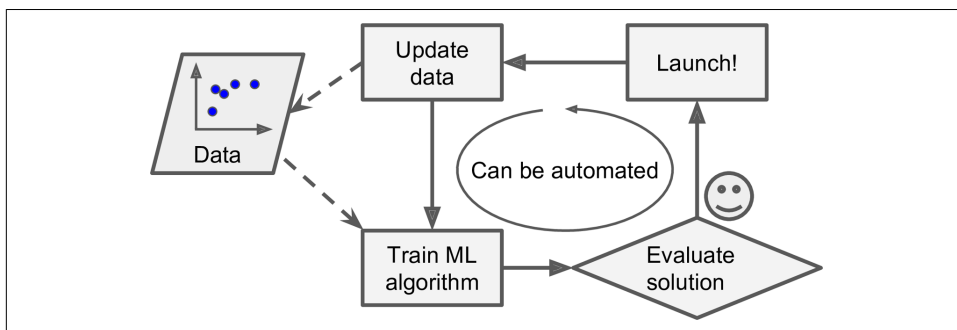


Figure 1-3. Automatically adapting to change

Another area where Machine Learning shines is for problems that either are too complex for traditional approaches or have no known algorithm. For example, consider speech recognition: say you want to start simple and write a program capable of distinguishing the words “one” and “two.” You might notice that the word “two” starts with a high-pitch sound (“T”), so you could hardcode an algorithm that measures high-pitch sound intensity and use that to distinguish ones and twos. Obviously this technique will not scale to thousands of words spoken by millions of very different people in noisy environments and in dozens of languages. The best solution (at least today) is to write an algorithm that learns by itself, given many example recordings for each word.

Finally, Machine Learning can help humans learn (Figure 1-4): ML algorithms can be inspected to see what they have learned (although for some algorithms this can be tricky). For instance, once the spam filter has been trained on enough spam, it can easily be inspected to reveal the list of words and combinations of words that it believes are the best predictors of spam. Sometimes this will reveal unsuspected correlations or new trends, and thereby lead to a better understanding of the problem.

Applying ML techniques to dig into large amounts of data can help discover patterns that were not immediately apparent. This is called *data mining*.

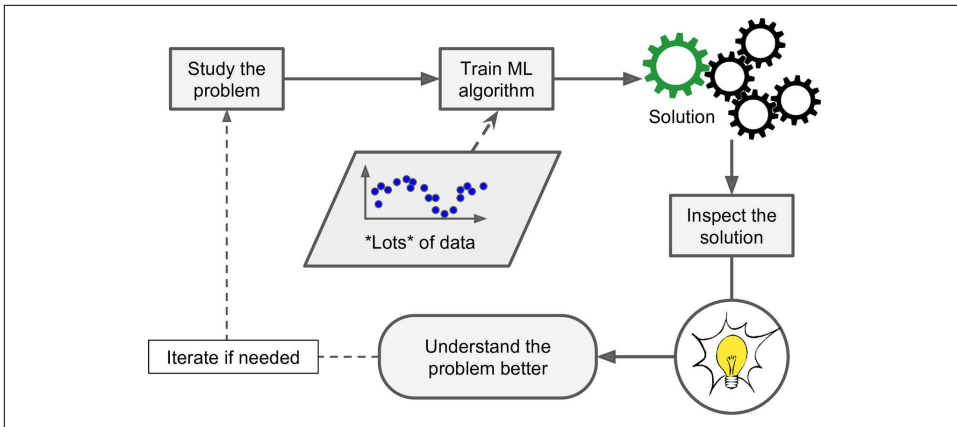


Figure 1-4. Machine Learning can help humans learn

To summarize, Machine Learning is great for:

- Problems for which existing solutions require a lot of hand-tuning or long lists of rules: one Machine Learning algorithm can often simplify code and perform better.
- Complex problems for which there is no good solution at all using a traditional approach: the best Machine Learning techniques can 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:

- Whether or not they are trained with human supervision (supervised, unsupervised, 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 detect patterns in the training data and build a predictive model, much like scientists do (instance-based versus model-based learning)

These criteria are not exclusive; you can combine them in any way you like. For example, a state-of-the-art spam filter may learn on the fly using a deep neural net-

visual cortex, 360  
visualization, 244-247  
visualization algorithms, 11-12  
voice recognition, 359  
voting classifiers, 183-186

## W

warmup phase, 354  
weak learners, 184  
weight-tying, 425  
weights, 269  
    freezing, 292  
while\_loop(), 393  
white box models, 172

worker, 328  
worker service, 330  
worker\_device, 332  
workspace directory, 40-43

## X

Xavier initialization, 278-281

## Y

YouTube, 255

## Z

zero padding, 362, 367

## About the Author

---

**Aurélien Geron** is a Machine Learning consultant. A former Googler, he led the YouTube video classification team from 2013 to 2016. He was also a founder and CTO of Wifirst from 2002 to 2012, a leading Wireless ISP in France; and a founder and CTO of Polyconseil in 2001, the firm that now manages the electric car sharing service Autolib’.

Before this he worked as an engineer in a variety of domains: finance (JP Morgan and Société Générale), defense (Canada’s DOD), and healthcare (blood transfusion). He published a few technical books (on C++, WiFi, and internet architectures), and was a Computer Science lecturer in a French engineering school.

A few fun facts: he taught his three children to count in binary with their fingers (up to 1023), he studied microbiology and evolutionary genetics before going into software engineering, and his parachute didn’t open on the second jump.

## Colophon

---

The animal on the cover of *Hands-On Machine Learning with Scikit-Learn and TensorFlow* is the far eastern fire salamander (*Salamandra infraimmaculata*), an amphibian found in the Middle East. They have black skin featuring large yellow spots on their back and head. These spots are a warning coloration meant to keep predators at bay. Full-grown salamanders can be over a foot in length.

Far eastern fire salamanders live in subtropical shrubland and forests near rivers or other freshwater bodies. They spend most of their life on land, but lay their eggs in the water. They subsist mostly on a diet of insects, worms, and small crustaceans, but occasionally eat other salamanders. Males of the species have been known to live up to 23 years, while females can live up to 21 years.

Although not yet endangered, the far eastern fire salamander population is in decline. Primary threats include damming of rivers (which disrupts the salamander’s breeding) and pollution. They are also threatened by the recent introduction of predatory fish, such as the mosquitofish. These fish were intended to control the mosquito population, but they also feed on young salamanders.

Many of the animals on O’Reilly covers are endangered; all of them are important to the world. To learn more about how you can help, go to [animals.oreilly.com](http://animals.oreilly.com).

The cover image is from *Wood’s Illustrated Natural History*. The cover fonts are URW Typewriter and Guardian Sans. The text font is Adobe Minion Pro; the heading font is Adobe Myriad Condensed; and the code font is Dalton Maag’s Ubuntu Mono.

**easy** ●●●  
**computing**