Hands-on Machine Learning

Theoretical Overview

Marc Evrard

2022-2023

Section 1

Course information

People

Lectures and practical works:

Marc Evrard (marc.evrard@lisn.upsaclay.fr)

Practical works:

Yue Ma (yue.ma@lisn.upsaclay.fr)

Registration:

Alexandre Verrecchia
(alexandre.verrecchia@universite-paris-saclay.fr)

References

The **main references** for this class:

- Géron (2019) *Hands-on Machine Learning with Scikit-Learn, Keras, and Tensorflow: Concepts, Tools, and Techniques to Build Intelligent Systems*
- *Scikit-learn User Guide*: https://scikit-learn.org/stable/user_guide.html
- VanderPlas (2017) *Python Data Science Handbook: Essential Tools for Working with Data.*

Great references for **machine learning algorithm** theory:

- Bishop (2006) *Pattern Recognition and Machine Learning*
- Russell and Norvig (2020) *Artificial Intelligence: A Modern Approach* (4th ed.)

Assessments (MCC)

- Continuous assessment (CC): 100%
  - Weekly quizzes (40%)
  - Practical assignments: 2 challenges (60%)
    - Prepared on Jupyter Notebooks
    - By teams of 2 or 3

Quizzes with degrees of confidence[†]

- Need to give a **degree of confidence**:
  Probability of the response you give is correct
- Self-assessing in a **realistic way** earns more points
- In general, the majority of students rate themselves well

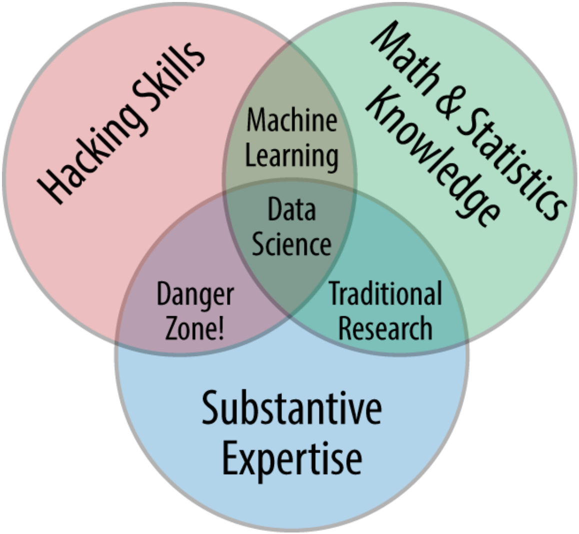
Probability	Degree	Correct	Wrong
0 to 25%	0	+13	+4
25 to 50%	1	+16	+3
50 to 70%	2	+17	+2
70 to 85%	3	+18	0
85 to 95%	4	+19	−6
95 to 100%	5	+20	−20

[†] http://smart.uliege.be

Section 2

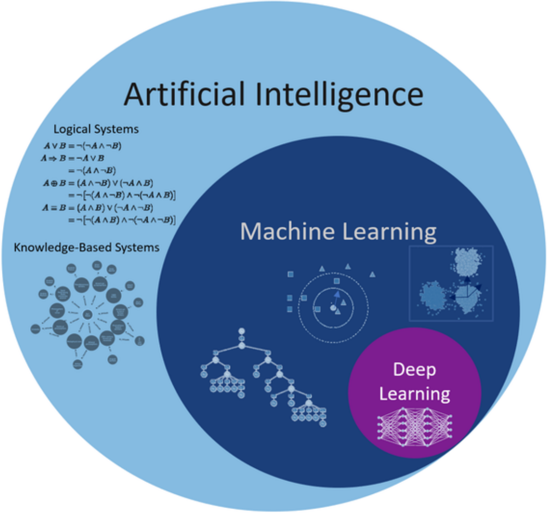
The Machine Learning Landscape

What is Machine Learning?†



† Drew Conway's Data Science Venn Diagram (from VanderPlas 2017)

What is Machine Learning?[†]



[†] Fig. from https://www.quora.com/profile/Jens-Laufer

What is Machine Learning?

- *Machine Learning is the science (and art) of programming computers so they can learn from data.* (Géron 2019)
- More generally:
  - *It's the field of study that gives computers the ability to* ***learn without being explicitly programmed.*** (Samuel 1959)
- More formally:

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.
(Mitchell 1997)

Spam filter example

- ML program that can learn to flag spam given examples of:
  - Spam emails (e.g., flagged by users)
  - Regular (non-spam) emails
- **Training set**: The examples used by the system to learn
  - **Training instance (or sample)**: Each training example
- In this case:
  - **Task T**: Flag spam for new emails
  - **Experience E**: Training data
  - **Performance measure P**:
    E.g., the ratio of correctly classified emails (accuracy)

Use of ML

ML is great for:

- Problems **too complex** to be solved by explicitly programming them
  - Require **complex rules**
  - Requiring a lot of **hands tuning**
- **Fluctuating environments**, which need to constantly adapt to new data
- **Getting insights** on complex problems and large amounts of data (data mining)

ML, the magical pill?

- The term *machine learning* is sometimes considered as a magical spell
  - *Apply ML to enough data, and any problems can be solved!*
- ML methods can be **extremely powerful**
  - But they are not equally capable of tackling all problems
- Several **issues** are highly likely to occur
  - Both on the data side and the ML methods side
  - **Bias** and **variance**, **overfitting** and **underfitting**, etc.

Types of ML Systems

According to the need for human supervision

Supervised:
The training data you feed to the algorithm includes the desired solutions (**labels**)

Unsupervised:
Modeling the variations of the features without reference to any labels

Semi-supervised:
Combines a small amount of labeled data with a large amount of unlabeled data during training

Reinforcement Learning:
Intelligent agents ought to take actions in an environment to maximize their cumulative reward

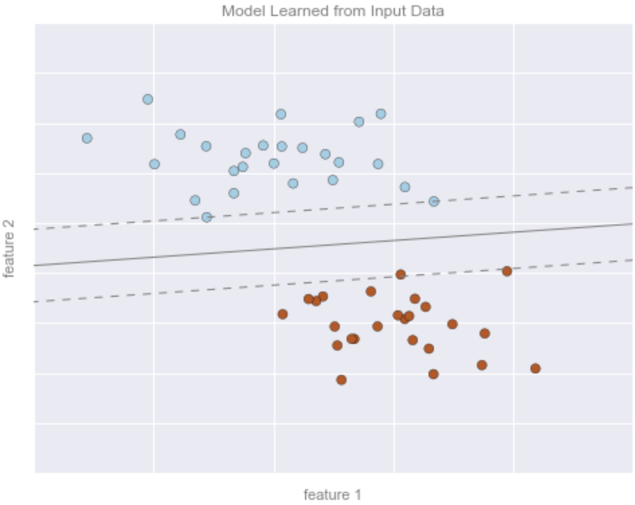
Supervised learning

- Supervised learning is further divided in:
  - **Classification** tasks
  - **Regression** tasks
- In a classification task the labels are **discrete categories**
  - Spam example: Predict the email tags (spam/ham)
- In a **regression** task the labels are **continuous quantities**: **Predicting a value** according to a set of features
  - Price of a house (features: area, size, age, condition, etc.)
- Regression algorithms can also be used for **classification**:
  - **Logistic Regression** is commonly used for classification
  - It can output a probability of belonging to a given class (e.g., 80% chance of being spam)

Supervised learning algorithms (main ones)

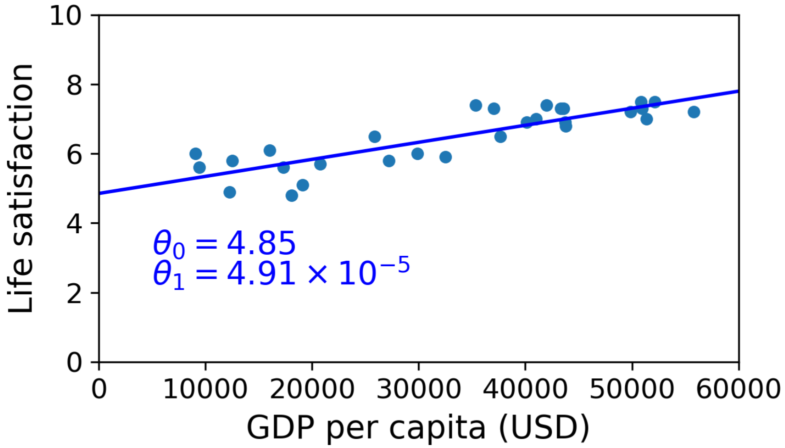
- **Classification**
  - k-Nearest Neighbors
  - Naive Bayes
  - Logistic Regression
  - Support Vector Machines (SVMs)
  - Decision Trees and Random Forests
- **Regression**
  - Linear Regression
  - Regression Trees
- **Both** classification and regression
  - Neural networks (NN)

Classification†



† Fig. from VanderPlas (2017)

Regression†



† Fig. from Géron (2019)

Unsupervised learning algorithms (main ones)

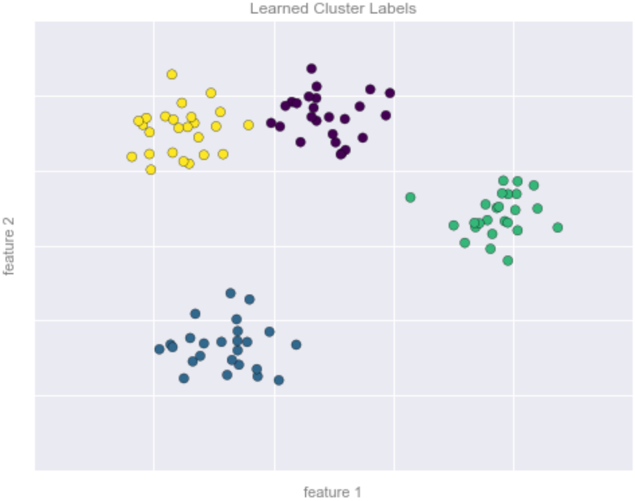
- **Clustering**
  - K-Means
  - Hierarchical Cluster Analysis (HCA)
- Visualization and **dimensionality reduction**
  - Principal Component Analysis (PCA) — Kernel PCA
  - t-distributed Stochastic Neighbor Embedding (t-SNE)
- **Self-supervised** (representation):
  - DNN: Autoencoders
  - Natural Language Processing (NLP):
    - Word embeddings
    - Transformers

Clustering I

Example: Large data about blog's visitors

- Run a clustering algorithm to try to **detect groups** of similar visitors
- You don't need to tell the algorithm which group a visitor belongs to
  - It finds those connections on its own but grouping them according to their **feature similarities**
  - E.g., it might notice that:
    - 40% of visitors love comic books and generally read the blog in the evening
    - 20% are young sci-fi lovers who visit during the weekends
    - ...

Clustering II†

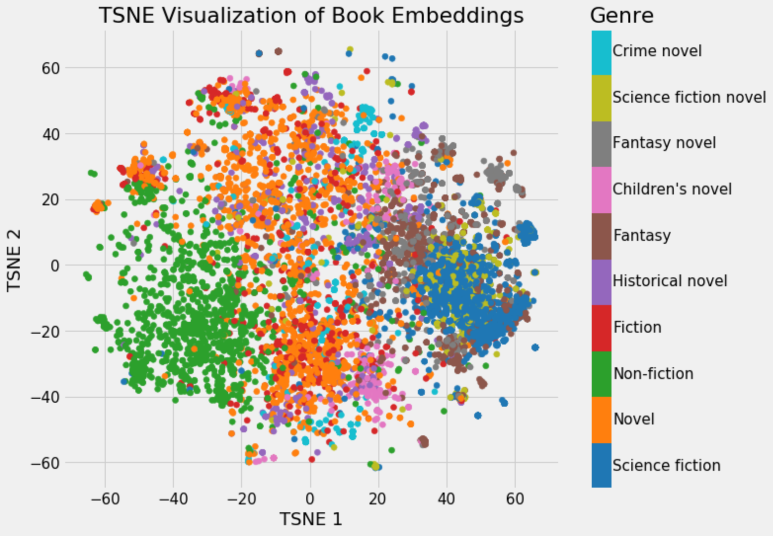


† Fig. from VanderPlas (2017)

Visualization algorithms

- Input large and **complex sets of unlabeled data**
- Output a **2D or 3D representation** of your data to be plotted
- Trying to **preserve** the **global** and **local structure** of the data:
  - Keep separate clusters in the input space from **overlapping** in the visualization
  - Help to identify **unexpected patterns**

t-SNE[†]



[†] From https://devopedia.org/word-embedding

Dimension reduction

- Task related to **visualization**
- **Goal**: **Simplify the data** without losing too much information
- **Merge** several **correlated features** into one
  - E.g., a car's mileage may be very correlated with its age
  - Merge them into one feature: car's wear and tear
  - **Feature extraction**
- Often a good idea to reduce the dimension of your **training**
  - Before feeding it to another Machine Learning algorithm (e.g., supervised learning algorithm)
    - Train much **faster**
    - Data needs **less disk** and **memory** space
    - **Perform better** in some cases

Batch vs Online learning

- Whether or not the system can learn incrementally from a stream of incoming data

Batch learning

- **Not incremental**: Trained using all the available data
- Time and computing **resource-consuming**:
  Need to be done offline
- System trained, then launched into production without learning anymore (**offline learning**)
- If new data needs to be included (e.g., new examples of spam), a new version must be **trained from scratch**

Batch vs Online learning

Online learning

- System trained **incrementally** by sequentially feeding data instances **individually** or by small groups: **Mini-batches**
- Each learning step is fast and cheap:
  The system can learn new data **on the fly**
- Ideal for systems that receive data as a **continuous flow** (e.g., stock prices)
- Good option when **limited computing resources** are available (e.g., smartphone)
- Solution for **out-of-core learning** from huge datasets:
  - Loads part of the data
  - Runs a training step sequentially on the whole corpus

Section 3

Main challenges

Main challenges of ML

- In short, tackling an ML problem supposes:
  - Selecting a **modeling algorithm**
  - That works best on **given data**
- The 2 main issues that thus can occur are:
  - **Algorithm** issues
  - **Data** issues
    - Insufficient **quantity** of training data
    - **Nonrepresentative** training data
    - Poor-**quality** data
    - Irrelevant **features**

Insufficient quantity of training data

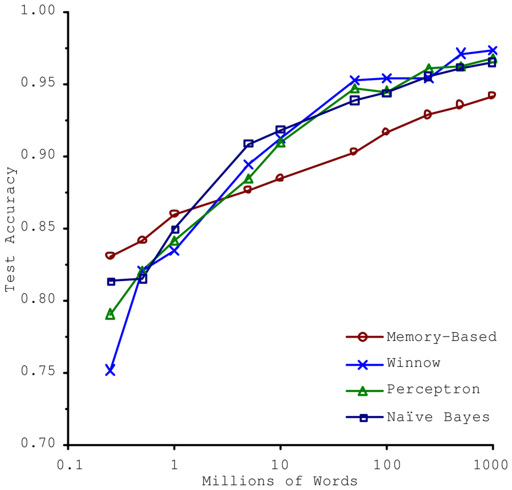
- To make a toddler learn what an apple is:
  Point to an apple and say "apple" (possibly a few times)
  - Now the child is able to recognize apples in all sorts of colors and shapes
- ML algorithms need a lot more data to achieve such results
  - Simple problems typically require thousands of examples
  - Complex problems (e.g., image or speech recognition tasks) may require millions of examples

    (Unless a semi-supervised approached is chosen or if an existing pretrained model is used through adaptation or fine-tuning)

The unreasonable effectiveness of data

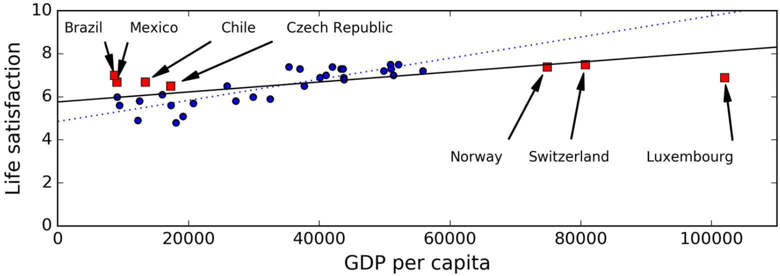
Microsoft researchers Banko and Brill (2001) showed (Fig.) that very **different algorithms** (including simple ones) performed almost **identically well** on complex problems provided **enough data** is given.

Further analyzed by Halevy, Norvig, and Pereira (2009) in *The unreasonable effectiveness of data.*



Nonrepresentative training data

- To **generalize well**: Training data must be **representative** of the **new cases**
- In the case of **linear regression**, some unseen data during training may not fit well with the predicted values
  - E.g., the GDP-Life Satisfaction (LS) relation:† It seems richest countries are not happier than moderately rich ones



†Fig. from Géron (2019)

Nonrepresentative training data: Sampling bias

- If the sample is **too small** $\Rightarrow$ sampling noise (i.e., high likelihood of nonrepresentative data)
- Even very large datasets can be nonrepresentative if the sampling method is biased
- Famous case of **sampling bias**:
  - US presidential election in 1936 (Landon vs Roosevelt)
  - Poll: **Mails** to about 10 million people (by Literary Digest)
  - They received 2.4 million answers:
    - Predicted Landon would win with 57%
    - Roosevelt won with 62% of the votes
  - Sampling issues:
    - **Exclusion bias**: subscribers, club membership lists, etc.
    - **Nonresponse bias** (25% answers)

Poor-quality data

- Plain errors: Outliers, and noise (e.g., from measurements)
- Important to **clean up** the training data
- **Significant part of the work** of researchers and data scientists
  - Remove **outliers**
  - If several instances are missing from some features:
    - Discard the attribute altogether
    - Ignore these instances
    - Fill in the missing values (e.g., with the median value)
    - A good idea is to **train multiple models** with the various possibilities and to compare their results

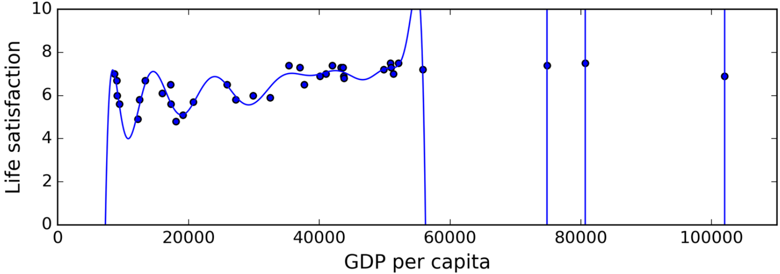
Irrelevant features

- *Garbage in, garbage out*
- Too many irrelevant features will prevent the system from learning efficiently
- A critical part of ML projects: **Feature engineering**
- Choosing a good set of features
  - **Feature selection**: Selecting the most useful features
  - **Feature extraction**: Combining existing features to produce the most useful ones

    (e.g., through dimensionality reduction like PCA)
  - Creating new features by gathering more data

Overfitting the training data

- One of the most striking examples of overfitting is perhaps xenophobia
- Like xenophobia, **overfitting** is the process of **overgeneralizing assumptions** (to be euphemistic)
- In ML: The model **performs well** on the **training** data, but **not** in the **test** data (it does not generalize well)[†]



[†]Fig. from Géron (2019)

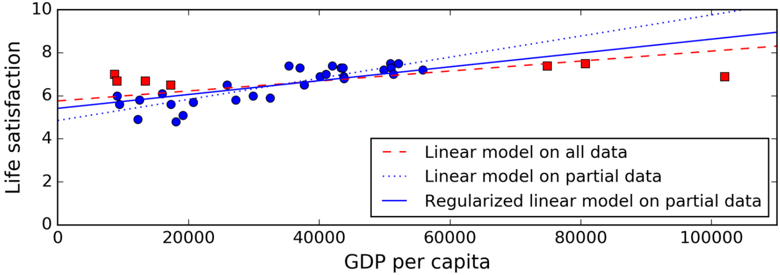
Overfitting the training data

- **Overfitting** is when the **model is too complex** relative to:
  - The **amount** of training data
  - The level of **noise** in the data
- The possible **solutions** are:
  - For the **model**:
    - Simplify the model by selecting one with **fewer parameters** (e.g., a linear model vs high-degree polynomial model)
    - **Constraining** the model
  - For the **data**:
    - **Reducing** the number of **features** in the training data
    - Gather **more training data**
    - **Reduce the noise** in the training data

Overfitting: Regularization

- **Regularization**: A way of **constraining** a model to make it less susceptible to overfitting
- E.g., The linear model: $LS = \theta_0 + \theta_1 \cdot GDP$
- It has 2 parameters: θ_0 and θ_1
  - 2 **degrees of freedom** to fit the model to the training data
  - Can adjust both the **height** (θ_0) and the **slope** (θ_1)
  - If we forced $\theta_1 = 0$: Only 1 degree of freedom
  - **Harder to fit** the data: Just a mean (very simple model indeed!)
  - If we force the model to keep it small: To be *in between* 1 and 2 degrees of freedom.

Overfitting: Regularization



- **Regularization** forced the model to have a smaller slope[†]
  - Fits less well the training data
  - But allows generalizing better to new examples
- The amount of regularization to apply during learning can be controlled by a **hyperparameter**

[†] Fig. from Géron (2019)

Underfitting the training data

- Underfitting occurs when your **model is too simple** to learn the underlying structure of the data
- E.g., the linear model of GDP-LS
- The possible **solutions** are:
  - Selecting a more powerful model (with more parameters)
  - **Feature engineering**
  - Reducing the constraints on the model (e.g., **reducing the regularization**)
- **Note**:
  - The previous slide illustrates the **overfitting** with the GDP-LS example ⇒ seems the **opposite!**
  - But a **reduced dataset** was given to the algorithm

Testing

- How well a model will **generalize** to new cases?

  Try on new cases

  - Try in production: Bad idea
  - Split the dataset into a **training** set and a **test** set
  - Common values: 80% data for training and 20% for testing

- **Generalization error**: The error rate on new cases

  - If the **training error** is **low**
  - And the **testing error** is **high**

    (High variance)

    $\Rightarrow$ **Overfitting**

Validating

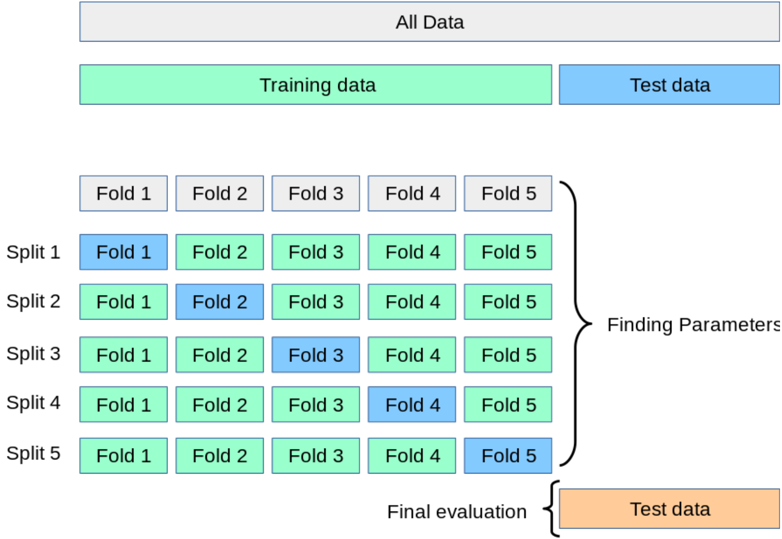
- If you want to test **multiple models**

  (e.g., a linear and polynomial model)
  - Train both and compare how they generalize on the test set
- You then want to apply some **regularization** to avoid overfitting
  - How do you choose the value of the regularization hyperparameter?
  - Train n different models using n different values for this hyperparameter
- Now the model and hyperparameters are the **best fit** for the test set
  - **Not likely to generalize** well on new data
  - Hold another set: The **validation set**

Cross-validation

- If the dataset is too small for a training/validation/test split: Use **cross-validation**
- The training set is split into **complementary subsets**
  - Each model is trained against a **different combination** of these subsets
  - And validated against the remaining parts
- Once the **model type** and **hyperparameters** have been selected
  - A final model is trained using these hyperparameters on the **full training set**
  - And the generalized error is measured on the **test set**

Cross-validation[†]



[†] From the Scikit-learn User Guide

No free lunch theorem

- A model is a **simplified version** of the observations
- The simplifications are meant to discard the superfluous details unlikely to generalize well
- **Assumptions** must be made to decide what data to discard/keep
- E.g., linear models suppose the assumption of **linearity** and that the rest is **noise**
- Wolpert and Macready (1997) demonstrated that:

  *If you make absolutely **no assumption** about the data, then there is **no reason** to prefer one model over any other*
- **No model is a priori guaranteed to work better**

Section 4

Bibliography

Bibliography I

Banko, Michele, and Eric Brill. 2001. "Scaling to Very Very Large Corpora for Natural Language Disambiguation." In *39th ACL*. Toulouse, France.

Bishop, Christopher M. 2006. *Pattern Recognition and Machine Learning*. Springer.

Géron, Aurélien. 2019. *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. 2nd ed. O'Reilly Media, Inc.

Halevy, Alon, Peter Norvig, and Fernando Pereira. 2009. "The Unreasonable Effectiveness of Data." *IEEE Intelligent Systems* 24 (2): 8–12.

Mitchell, Tom. 1997. "Machine Learning."

Russell, Stuart J, and Peter Norvig. 2020. *Artificial Intelligence: A Modern Approach*. 4th ed. Pearson Education.

Bibliography II

Samuel, Arthur L. 1959. "Some Studies in Machine Learning Using the Game of Checkers." *IBM Journal of Research and Development* 3 (3): 210–29.

VanderPlas, Jake. 2017. *Python Data Science Handbook: Essential Tools for Working with Data*. O'Reilly Media, Inc.

Wolpert, David H, and William G Macready. 1997. "No Free Lunch Theorems for Optimization." *IEEE Transactions on Evolutionary Computation* 1 (1): 67–82.