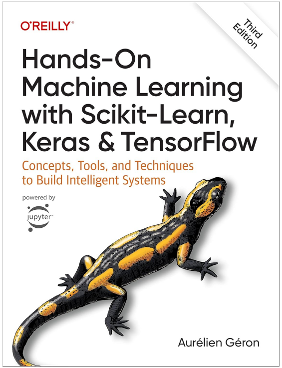
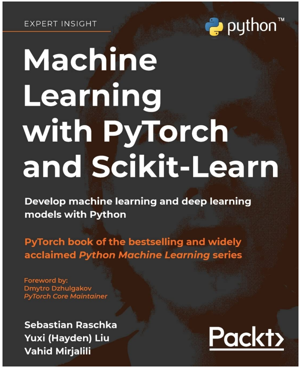
THE MACHINE LEARNING PROBLEM

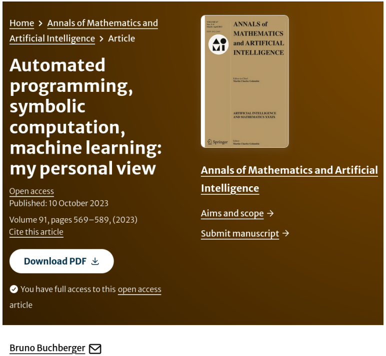
A Bird's-Eye View

Wolfgang Schreiner
Research Institute for Symbolic Computation (RISC)
Johannes Kepler University Linz, Austria

Sources







What is the core problem solved by ML and what is the basic solution strategy?

The ML Problem

Automatically generate a program that solves a problem specified by (input, output) examples.

- **Given:** a sequence $T \in Pr^* \subseteq (I \times O)^*$ of (input, output) examples
  - Pr: the *problem*, i.e., the set of all legal (input, output) pairs from domains I and O

  * such that
  - T (the *training set*) is "representative" for Pr.
- **Find:** a *model* M
  - A finitary representation of a function $[\![M]\!] : I \to O$

  such that M makes "good" predictions for the inputs in T, i.e., we have

  $$P(M^*(T)) > p.$$

  - $P : (O \times O)^* \to \mathbb{R}$: the *performance measure*, $p \in \mathbb{R}$: the *desired performance*
  - $M^*(T) := [(\hat{y}, y) \mid (x, y) \leftarrow T, \hat{y} = [\![M]\!](x)]$
    - x: the *input*, y: the *output* (*target*), $\hat{y}$: the *prediction*.
- **Test:** for candidate M, choose *test set* $U \in Pr^*$ "disjoint" from T and check $P(M^*(U)) > p$:
  - If the check succeeds, then M "generalizes" to the test set and is accepted.
  - If not, then M "overfits" the training set and is rejected.

If the model generalizes to the test set, it *may* also generalize to the full problem.

The ML Meta-Problem

To solve the ML problem, we typically have to solve the following “meta-problem”.

- **Given:** $T \in Pr^* \subseteq (I \times O)^*$.
- **Find:** a *model (template)* $MT^{hp}[\theta]$, values $\overline{hp}$ and $\bar{\theta}$ for its *hyperparameters* and *parameters*.
  - $hp \in HP^*$: the model *hyperparameters*.
  - $\theta \in \mathbb{R}^*$: the (numerical) model *parameters* (*weights*).

  such that we have $P(M^*(T)) > p$ where
  - $\overline{MT}[\theta] = MT^{\overline{hp}}[\theta]$
  - $M = \overline{MT}[\bar{\theta}]$

We have to find a suitable model (template), suitable values for its hyperparameters, and suitable values for its parameters.

The ML Meta-Problem (Refined)

The problem of finding suitable values for the model parameters can be framed as a problem of numerical optimization.

- **Given:** $T \in Pr^* \subseteq (I \times O)^*$.
- **Find:** a *model (template)* $MT^{hp}[\theta]$, values $\overline{hp}$ for its *hyperparameters*, and a *loss function* L.
  - $L\colon (O \times O)^* \to \mathbb{R}$: maps a list of pairs $(\hat{y}, y)$ to a numerical *loss* (*cost*, *error*).
  - Strongly correlated (but not necessarily identical) to the negation of P.

  such that we have $P(M^*(T)) > p$ where
  - $\overline{MT}[\theta] = MT^{\overline{hp}}[\theta]$
  - $\bar{\theta}$ is a value for θ that minimizes $L[(\bar{y}, y) \mid (x, y) \leftarrow T, \bar{y} = [\![\overline{MT}[\theta]]\!](x)]$
  - $M = \overline{MT}[\bar{\theta}]$

We have to select a suitable loss function and minimize it.

The ML Meta-Problem (Training/Model Fitting)

But then we also have to decide how to solve the minimization problem.

- **Given:** $T \in Pr^* \subseteq (I \times O)^*$.
- **Find:** a *model (template)* $MT^{hpm}[\theta]$, values $\overline{hpm}$ for its hyperparameters, a *loss function* L, a *training algorithm ("optimizer")* TA^{hpt}, and values $\overline{hpt}$ for its hyperparameters
  - TA^{hpt}: a function that (approximately) solves the minimization problem.

  such that we have $P(M^*(T)) > p$ where
  - $\overline{MT}[\theta] = MT^{\overline{hpm}}[\theta]$
  - $\bar{\theta} = TA^{\overline{hpt}}(\overline{MT}[\theta], L, T)$ ("training the model")
  - $M = \overline{MT}[\bar{\theta}]$

We have to select an appropriate algorithm for solving the minimization problem and suitable values for its hyperparameters.

The ML Meta-Problem (Validation/Hyperparameter Tuning)

It remains to choose suitable values $\overline{hpm}, \overline{hpt}, \ldots$ such that the resulting model most likely generalizes well to the test set (and the full problem)...

- **Given:** $T \in Pr^* \subseteq (I \times O)^*$
- **Find:** model M such that $P(M^*(T)) > p$
  - Let V be some “part” of T and let T' be T “without” V.
    - V: the *validation set*.
  - Choose as $\overline{hpm}, \overline{hpt}, \ldots$ values for $hpm, hpt, \ldots$ (from a set of candidates) that maximize

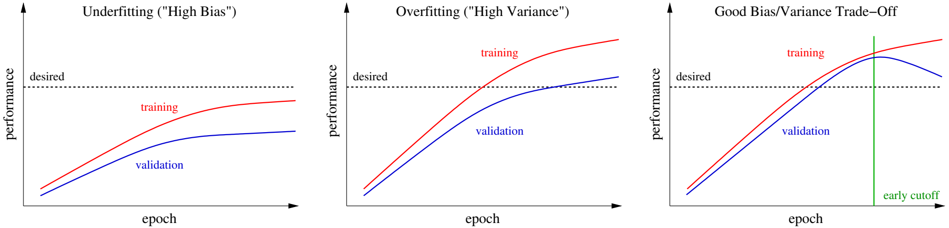
    $$P[(\bar{y}, y) \mid (x, y) \leftarrow V, \bar{y} = [\![\overline{MT}[\bar{\theta}]]\!](x)]$$ (validating the model on V)

    where
    - $\overline{MT}[\theta] = MT^{hpm}[\theta]$
    - $\bar{\theta} = TA^{hpt}(\overline{MT}[\theta], L, T')$ (training the model on T')
  - Let $M = \overline{MT}[\bar{\theta}]$
    - $\overline{MT}$ and $\bar{\theta}$ are determined by $\overline{hpm}, \overline{hpt}, \ldots$.

We generate from the training set models for various hyperparameter combinations (“grid/randomized search”) and select the one that generalizes best to the validation set.

Evaluating the Model Performance

When training the model, the optimizer runs over the training set a certain number of times ("epochs"); how do we know when the the model is adequate or can/should be further improved my more training?



Learning curves can be used to judge the adequacy of the model.

Summary

To develop an adequate machine learning model, we have to

- choose a model template,
- an optimizer,
- hyperparameters for model and optimizer,
- a loss function;
- train the model using the optimizer and loss function;
- evaluate the performance of the trained model;
- repeat the process until we have a model that neither underfits nor overfits.

Many other topics: collection and preparation of data, labeling data, model types, neural networks (architectures and training), reuse of models (transfer learning), reinforcement learning, ...