Interpretable Machine Learning: On the Problem of Explaining Model Change

Barbara Hammer[1], Eyke Hüllermeier[2]

[1]Faculty of Technology
Bielefeld University
E-Mail: bhammer@techfak.uni-bielefeld.de

[2]Institute of Informatics
LMU Munich
E-Mail: eyke@lmu.de

1 Introduction

Over the past couple of years, the idea of explainability and related notions such as transparency and interpretability have received increasing attention in artificial intelligence (AI) in general and machine learning (ML) in particular. This is mainly due to the ever growing number of real-world applications of AI technology and the increasing level of autonomy of algorithms taking decisions on behalf of people, and hence of the social responsibility of computer scientists developing these algorithms. Recent methods for improving the understandability and transparency of models produced by ML algorithms include both model-specific [6] as well model-agnostic approaches [12].

These approaches have largely focused on the explanation of *static* models, typically learned on a set of training data in a batch mode. Arguably more challenging is interpretability in the context of learning in non-stationary environments, where models are learned on a continuously evolving, potentially unbounded stream of temporally ordered data, and incrementally updated in the light of newly observed training examples [5, 4]. Corresponding algorithms must be able to react to changes in the underlying data-generating process,

DOI: 10.58895/ksp/1000138532-1 erschienen in:
Proceedings - 31. Workshop Computational Intelligence : Berlin, 25. - 26. November 2021
DOI: 10.58895/ksp/1000138532 | https://www.ksp.kit.edu/site/books/m/10.58895/ksp/1000138532/

which is referred to as *concept drift* [10, 8]. Concept drift may call for incremental adaptations and sometimes also more significant modifications of the model — in the extreme case of an abrupt change, the learner may even decide to abandon the current model completely and start learning from scratch.

Explaining model change, whether incremental or abrupt, is important in practical applications of online learning and can be seen as a key prerequisite for user acceptance. In particular, it is well known that humans prefer stability to change [2] — they tend to rely on what is predictable from the past and are cognitively challenged by deviations from an established solution. Hence, a good explanation is required to convince a user of any need for changing the current model.

Taking the stance that this explanation should focus on the change itself, that is, on the *differences* between the original and the updated model, we subsequently elaborate on the idea of *explaining model change* and identify a number of important problems to be addressed in this regard. In Section 3, we illustrate these problems for the specific example of instance-based learning on data streams.

2 Explaining Model Change

Consider a sequence of models $(h_t)_{t\in T}$ produced by an incremental learning algorithm A, where $T \subset [0,\infty)$ is a countable set of time indices, for example $T = \mathbb{N}$. The model $h_t : \mathscr{X} \longrightarrow \mathscr{Y}$ is produced on the basis of the data

$$\mathscr{D}_t = \big\{(x_i, y_i)\big\}_{i\in T\cap[0,t]} \subset \mathscr{X} \times \mathscr{Y}$$

observed by the learner till time t, where $\mathscr{X}$ and $\mathscr{Y}$ denote the underlying instance and outcome space, respectively (cf. Fig. 1 for an illustration). The data generating process is characterized by a corresponding sequence of probability distributions $(P_t)_{t\in T}$ on $\mathscr{X} \times \mathscr{Y}$, which may evolve over time (i.e., $P_s \neq P_t$ for $s \neq t$) and, of course, is not known to the learner; thus, we assume that each data point (x_t, y_t) is generated by P_t [16, 9].

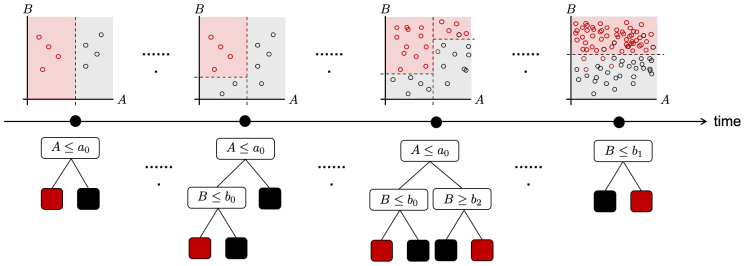

Figure 1: Illustration of model change (here for the case of decision trees) over the course of an incremental learning process.

If data and models evolve quickly, perhaps even in realtime, it will not be possible to explain every single model h_t to a user or human domain expert. Besides, individual explanations of that kind, isolated from each other, might be problematic for the user anyway, especially in the case of inconsistencies. Instead, the user might be more interested in how the model *changes* over the course of time, and in understanding the reasons for these changes. This gives rise to the idea of explaining model change in the sense of the "difference" between models, a task that appears to be more feasible, especially if changes are local, i.e., restricted to certain parts of a model or a local region of $\mathscr{X}$.

More concretely, consider a scenario in which, at every time point $t \in T$, the user has information about a previous model h_{t_0}, where $t_0 < t$. This *reference model* is not necessarily up to date, because the learning process has progressed since then and produced updated models $(h_s)_{s \in T \cap (t_0,t]}$. What we mean by explaining a model change is to inform the user about the "difference" $\Delta(h_{t_0}, h_t)$ between the reference and the current model and making h_t the new reference. Questions, problems, and challenges arising in this context include the following:

Q1 What are suitable representations of models and model change?

Q2 How to quantify model change, i.e., the difference $\Delta(h_{t_0}, h_t)$ between models h_t and h_{t_0} (distinguishing between *syntactic* difference referring to the representation of a model and *semantic* difference referring to the change of the functional dependence $\mathscr{X} \longrightarrow \mathscr{Y}$)?

Q3 How to compute $\Delta(h_{t_0}, h_t)$ efficiently, preferably in an incremental manner?

Q4 When and how often should a model change be explained, bearing in mind aspects of computational complexity, but perhaps more importantly the cognitive capacity of the human user (who is likely to prefer stability over change)?

Q5 How to complement the explanation of a change by convincing reasons for why it was needed?

Obviously, suitable answers to these questions will strongly depend on the learning task and the type of model produced by the learning algorithm. In the next section, we illustrate the problems for a specifically simple example, namely, the case of instance-based learning on data streams.

3 Instance-Based Learning on Data Streams

The notion of instance-based learning (IBL) refers to a family of machine learning algorithms, including memory-based learning, exemplar-based learning, and case-based learning [13, 7], which represent a predictive model in an indirect way via a set of stored data. Thus, in contrast to model-based machine learning methods which induce a general model (theory) from the data and use that model for further reasoning, IBL algorithms simply store the data itself and defer its processing until a prediction (or some other type of query) is actually requested — a property which qualifies them as a *lazy* learning method [1]. Predictions are then derived by combining the information provided by the stored examples, typically accomplished by means of the nearest neighbor (NN) estimation principle [3]. In this regard, examples are also referred to as *cases*, and the stored data as the *case base*.

More specifically, consider the simple example of binary classification with data of the form $\mathscr{D} = \{(x_i, y_i)\}_{i=1}^N \subset \mathscr{X} \times \mathscr{Y}$, where $\mathscr{X} = \mathbb{R}^d$ and $\mathscr{Y} = \{0, 1\}$. The instance space $\mathscr{X}$ is equipped with a distance measure, for example the

Euclidean metric. Adopting the simple nearest neighbor rule, the model $h_{\mathscr{D}}$ induced by the data $\mathscr{D}$ is given by

$$h_{\mathscr{D}} : \mathscr{X} \to \mathscr{Y}, x \mapsto y_{\mathrm{NN}(x,\mathscr{D})},$$

where $\mathrm{NN}(x,\mathscr{D})$ denotes the (index of the) nearest neighbor[1] of x in $\mathscr{D}$, i.e.,

$$\mathrm{NN}(x,\mathscr{D}) = \arg\min_{1\leq i\leq N} \|x - x_i\|.$$

Obviously, using $h_{\mathscr{D}}$ to make predictions for new query instances x_q requires searching for the nearest neighbor of x_q in the data $\mathscr{D}$. Although the complexity of nearest neighbor search can be reduced by means of specific data structures [11], instance-based learning will typically remain more costly at prediction time than model-based learning.

On the other side, an instance-based approach naturally supports an incremental mode of learning. In fact, in the data stream scenario, where new cases are observed continuously over the course of time, the problem of learning essentially reduces to the problem of *case based editing* or *case based maintenance* [15]: every time a new example (x_{new}, y_{new}) arrives, one needs to decide whether or not this example should be added to $\mathscr{D}$, and if other cases should perhaps be removed. Disregarding computational complexity, the ideal case base $\mathscr{D}^* \subseteq \mathscr{D} \cup \{(x_{new}, y_{new})\}$ will maximize predictive performance (classification accuracy) of the induced classifier in the future. As this criterion cannot be used directly (future performance is difficult to anticipate, especially in the presence of concept drift), most methods fall back on suitable indicators of the usefulness of individual cases. For example, the IBLStreams approach [14] decides about the addition or removal of cases on the basis of the following criteria:

- Temporal relevance: Recent observations are deemed potentially more useful and are hence preferred to older ones.
- Spatial relevance: Examples can be redundant in the sense of not changing the nearest neighbor classification of any query. More generally (and less stringently), one might consider a set of examples redundant

[1] A tie breaking mechanism is needed in the case where the nearest neighbor is not unique.

if they are closely neighbored in the instance space and, hence, have a similar region of influence (Voronoi cell). In other words, a new example in a region of the instance space already occupied by many other examples is considered less relevant than a new example in a sparsely covered region.

- Consistency: An example should be removed if it seems to be inconsistent with the current concept, e.g., if its class label differs from most of the labels in its neighborhood. In this regard, however, it is important to distinguish between "noisy cases" and the possible beginning of a concept drift.

Bringing the aspect of explainability into play, we can imagine a learner adopting principles of this kind to edit its case base but delaying the update. In other words, the learner maintains a *candidate* case base $\mathscr{D}_t$ in parallel to the *reference* case base $\mathscr{D}_{t_0}$ that is used to make predictions. Thus, whenever $\mathscr{D}_t$ is modified, the learner has to decide whether to retain $\mathscr{D}_{t_0}$ or replace it by $\mathscr{D}_t$. Let us reconsider the questions Q1–Q5 for this particular scenario.

As for Q1, we already mentioned that models are represented indirectly in instance-based learning: a model $h_{\mathscr{D}_{t_0}}$ is represented by a set of cases $(x_i, y_i) \in \mathscr{D}_{t_0}$, which can be presented to a user as prototypical examples. Seen from this perspective, the case base should be kept as small as possible, because overly large case bases will compromise interpretability. Individual predictions $h_{\mathscr{D}_{t_0}}(x_q)$ are naturally "justified" by means of similarity-based or example-based explanations referring to local (nearest neighbor) information in the vicinity of the query x_q. In the simplest case, the nearest neighbor is retrieved and its class label is provided as a justification: "There is a case x_i that belongs to class y_i and resembles x_q, so x_q is likely to belong to y_i as well."

As for Q2, the syntactic difference between $h_{\mathscr{D}_{t_0}}$ and $h_{\mathscr{D}_t}$ is naturally defined in terms of the (cardinality of the) symmetric difference $(\mathscr{D}_{t_0} \cup \mathscr{D}_t) \setminus (\mathscr{D}_{t_0} \cap \mathscr{D}_t)$ between $\mathscr{D}_{t_0}$ and $\mathscr{D}_t$. Likewise, a natural definition of the semantic difference is the expected discrepancy

$$\Delta(h_{\mathscr{D}_{t_0}}, h_{\mathscr{D}_t}) = \int_{\mathscr{X}} \| h_{\mathscr{D}_{t_0}}(x) - h_{\mathscr{D}_t}(x) \| \, p(x) dx,$$

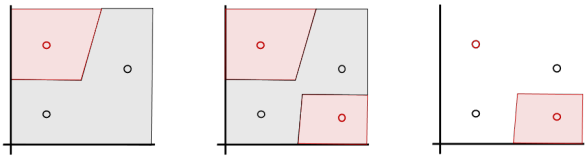

Figure 2: Illustration of a model change in the case of nearest neighbor classification. A model is characterized by a Voronoi tessellation. Adding a new example to the original model (left) leads to a change of the model (middle) and a corresponding difference (right) to be explained to the user.

where $p(x)$ is the probability (density) of observing x as a query. Because the latter is not known and difficult to estimate, especially in the presence of concept drift, one may think of

$$\Delta(h_{\mathscr{D}_{t_0}}, h_{\mathscr{D}_t}) = \int_{\mathscr{X}} \|h_{\mathscr{D}_{t_0}}(x) - h_{\mathscr{D}_t}(x)\| \, dx \tag{1}$$

as an alternative, effectively assuming a uniform distribution on $\mathscr{X}$. Obviously, $\mathscr{X}$ must be bounded in this case, which can be guaranteed through normalization, for example by mapping $\mathscr{X}$ to $[0,1]^d$; a transformation of this kind is anyway advisable to assure commensurability between the different features (dimensions) and hence the meaningfulness of the Euclidean metric.

Turning to Q3, the computation of (1) is an algorithmically challenging problem, which comes down to identifying the (volume of) the *discrepancy region* in $\mathscr{X}$, *viz.* the set of points x for which the label of the nearest neighbor in $\mathscr{D}_{t_0}$ differs from the label of the nearest neighbor in $\mathscr{D}_t$ (cf. Fig. 2). While an efficient algorithmic solution to this problem is beyond the scope of this paper, we mention that a simple approximation can be obtained through Monte Carlo sampling:

$$\Delta(h_{\mathscr{D}_{t_0}}, h_{\mathscr{D}_t}) \approx \frac{1}{K} \sum_{k=1}^{K} \|h_{\mathscr{D}_{t_0}}(x'_k) - h_{\mathscr{D}_t}(x'_k)\| \,,$$

where $x'_1, \ldots, x'_K$ are sampled uniformaly at random from $\mathscr{X}$.

As for Q4 and Q5, the learner needs to take both $\Delta(h_{\mathscr{D}_{t_0}}, h_{\mathscr{D}_t})$ and the difference between $h_{\mathscr{D}_t}$ and $h_{\mathscr{D}_{t_0}}$ in terms of (estimated) usefulness into account. The

larger these quantities, the stronger the need for an update. The explanation of an update then essentially comes down to informing the user about the symmetric difference between the corresponding cases bases, i.e., explaining that some of the previous cases have become redundant or are no longer considered sufficiently prototypical, while other cases have been added as new prototypes. To convince the user of the need for a revision of the case base, one may present examples of queries that are classified correctly with the new model but incorrectly with the old one.

4 Conclusion

We motivated the task of explaining the change of models in the context of learning in dynamic environments, where data is coming in streams and continuously evolving over the course of time, possibly urging the learner might to react to concept drift. In this regard, we highlighted a number of problems and challenges to be addressed, and illustrated these problems for the specific case of instance-based learning on data streams. As a next step, we seek to realize these ideas on a more technical level, put them into practice, and evaluate them in the context of real applications. Besides, we shall study the problem of explaining model change also for other learning tasks and other model classes.

References

[1] D.W. Aha, editor. *Lazy Learning*. Kluwer Academic Publ., 1997.

[2] A. Clark. Whatever next? Predictive brains, situated agents, and the future of cognitive science. *Behavioral and Brain Sciences*, 36(3):181–204, 2013.

[3] B.V. Dasarathy, editor. *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*. IEEE Computer Society Press, Los Alamitos, California, 1991.

[4] G. Ditzler, M. Roveri, C. Alippi, and R. Polikar. Learning in nonstationary environments: A survey. *IEEE Computational Intelligence Magazine*, 10(4):12–25, 2015.

[5] G. Morales De Francisci and A. Bifet. SAMOA: scalable advanced massive online analysis. *Journal of Machine Learning Research*, 16(1):149–153, 2015.

[6] J.H. Friedman and B.E. Popescu. Predictive learning via rule ensembles. *The Annals of Applied Statistics*, 2(3):916–954, 2008.

[7] J.L. Kolodner. *Case-based Reasoning*. Morgan Kaufmann, San Mateo, 1993.

[8] V. Losing, B. Hammer, and H. Wersing. KNN classifier with self adjusting memory for heterogeneous concept drift. In *IEEE International Conference on Data Mining (ICDM)*, pages 291–300, 2016.

[9] V. Losing, B. Hammer, and H. Wersing. Incremental on-line learning: A review and comparison of state of the art algorithms. *Neurocomputing*, 275:1261–1274, 2018.

[10] V. Losing, B. Hammer, and H. Wersing. Tackling heterogeneous concept drift with the Self-Adjusting Memory (SAM). *Knowledge and Information Systems*, 54(1):171–201, January 2018.

[11] Y. Malkov and D. Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *CoRR*, abs/1603.09320, 2016.

[12] M.T. Ribeiro, S. Singh, and C. Guestrin. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Proc. 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 1135–1144. ACM Press, 2016.

[13] S. Salzberg. A nearest hyperrectangle learning method. *Machine Learning*, 6:251–276, 1991.

[14] A. Shaker and E. Hüllermeier. IBLStreams: a system for instance-based classification and regression on data streams. *Evolving Systems*, 3(4):235–249, 2012.

[15] B. Smyth and E. McKenna. Competence models and the maintenance problem. *Computational Intelligence*, 17(2):235–249, 2001.

[16] G.I. Webb, L.K. Lee, F. Petitjean, and B. Goethals. Understanding concept drift. *arXiv preprint arXiv:1704.00362*, 2017.