Bandits for Learning to Explain from Explanations

Freya Behrens,[1] **Stefano Teso,**[2] **Davide Mottin**[3]
[1] Technische Universität Berlin, Germany
[2] University of Trento, Italy
[3] Aarhus University, Denmark
f.behrens@campus.tu-berlin.de, stefano.teso@unitn.it, davide@cs.au.dk

Abstract

We introduce EXPLEARN, an online algorithm that learns to jointly output predictions *and* explanations for those predictions. EXPLEARN leverages Gaussian Processes (GP)-based contextual bandits. This brings two key benefits. First, GPs naturally capture different kinds of explanations and enable the system designer to control how explanations generalize across the space by virtue of choosing a suitable kernel. Second, EXPLEARN builds on recent results in contextual bandits which guarantee convergence with high probability. Our initial experiments hint at the promise of the approach.

Introduction

Nowadays, a physician who needs to discriminate between sick and healthy patients might use a black-box classifier to support her decisions. If the classifier predicts a serious diagnosis for which invasive treatment is necessary, should the physician trust its prediction?

Tools from explanatory machine learning (**?**) help decision makers, like the physician in our example, to understand the predictions of black-box models. In the simplest case, these so-called *explainers* unpack the reasons behind individual predictions by identifying those "explanatory features"—namely input variables like pixels and words, or higher-level concepts like interpretable image segments—that are most responsible for a given prediction (**?**). Explainers, albeit useful in evaluating the model, do not directly allow the users to *correct* the model's bugs. In our example, even if the physician noticed that the classifier's decision depends on image artifacts in a lung ultrasound, she could do little to correct the reasoning of the predictor.

As a remedy to this lack of control, we propose EXPLEARN, a principled and general approach for learning models that *explain their own predictions* and that *learn from supervision about their own explanations.*

EXPLEARN combines in an original way two key ingredients. First, the flexibility of Gaussian Processes (GP) (**?**), a class of non-parametric kernel-based models. Contrary to existing techniques (**???**) that are restricted to a single type of explanations, our approach accommodates different kinds of explanations by virtue of kernels. Second, EXPLEARN frames the interaction with the user in terms of contextual bandits and leverages principled algorithms for GP-based bandits (**?**) that guarantee convergence with high probability to a model that outputs high-quality predictions *and* explanations, as long as such a model exists.

An important aspect of EXPLEARN is that it learns models that are self-explainable (**?**) since a prediction is always accompanied by a corresponding explanation, without any extra steps. This is in sharp contrast with standard approaches, such as LIME (**?**), that extract approximate explanations using model distillation (**?**).

In summary, we:

- Introduce EXPLEARN, an approach that combines Gaussian Processes and contextual bandits for learning self-explainable models by interacting with a domain expert.
- Discuss the conditions under which our learning task is well-founded, and in particular what kernel functions are most appropriate.
- Present initial evidence that EXPLEARN learns good self-explainable models on a real-world and on a non-trivial synthetic dataset.

Problem Statement and Conceptualization

We tackle *learning to explain* from ground-truth explanations in an interactive setting.

Example. *Consider a physician who diagnoses sick or healthy patients and relies on a classifier to support the decisions. The classifier analyzes data about each patient and returns a prediction and an explanation. With such information, the doctor can decide whether to trust the prediction.*

Example. *A fake news detector is a model that predicts the truthfulness of an article and at the same time why the article has a certain rating. Based on this information, a journalist can evaluate the reliability of the prediction.*

In such critical scenarios, we require that the reasons behind the machine's predictions are always available to the user, who, in turn, should be allowed to correct the model whenever the predictions or explanations are wrong.

Since explanations and labels are neither readily available nor easy to access, we cast these scenarios in terms of sequential learning. At each iteration $t = 1, 2, \dots, T$, the machine receives an instance $x_t \in \mathcal{X} := \mathbb{R}^n$ to be classified (the patient status) and returns both a *prediction* $\hat{y}_t \in \mathcal{Y}$ (a diagnosis) and an *explanation* $\hat{z}_t \in \mathcal{Z}$ for the prediction (symptoms that support the diagnosis). A human expert then provides feedback on the prediction and on how well the explanation does support the prediction. Based on this feedback, the machine aims to learn a model that returns accurate predictions and explanations on future instances.

A Contextual Bandits Formulation

Our first contribution is a formulation of this problem in terms of *contextual bandits*, a principled theory for learning and optimization (**??**), in which a learner observes a context, pulls an arm based on the context, and receives a reward. In our case, instances x_t are contexts and prediction-explanation pairs $(\hat{z}_t, \hat{y}_t)$ are arms. Upon playing an arm, corresponding to making a prediction, the machine receives a reward $f_t := f(x_t, \hat{z}_t, \hat{y}_t) + \epsilon_t$ from the user, where ϵ_t is a noise term. The reward is proportional to the quality of the prediction and explanation. Learning then amounts to estimating f from the observed rewards. The main challenge is how to balance *exploitation* of high-quality arms and *exploration* of arms with high information for learning.

Contextual bandits measure the performance with *contextual regret* (or simply regret), which represents the reward lost by playing the predicted arm $(\hat{z}_t, \hat{y}_t)$ instead of the best possible arm. Regret is defined as:

$$r_t = \sup_{z,y} f(x_t, z, y) - f(x_t, \hat{z}_t, \hat{y}_t) \tag{1}$$

The cumulative regret R_T of the machine in T rounds is:

$$R_T = \sum_{t=1}^{T} r_t \tag{2}$$

A major advantage of this formulation is that it enables us to choose from several algorithms for contextual bandits that *provably* minimize the cumulative regret and, in doing so, quickly learn how to choose high-reward arms. Such guarantees are critical in our scenario, where explanations and labels are scarce and user interaction expensive.

Challenges

Effective strategies for learning to explain have to account for a number of challenges, which we briefly outline:

1. **Control.** With no supervision on the explanations, the machine might produce nonsensical or biased explanations (**???**). Hence, supervision and ground-truth explanations are necessary.
2. **Flexibility.** No single definition of explanation exists (**??**). Different forms of explanations have been identified, e.g., relevant features (**?**), influential training examples (**?**), counterfactual configurations (**?**), and combinations of multiple approaches (**?**). A proper strategy for learning to explain should accommodate different forms of explanations based on the application and the inclinations of the user. We discuss below a list of candidate explanations for our approach.
3. **Consistency.** There is a tight coupling between predictions and explanations. For this reason, the instance x_t, prediction $\hat{y}_t$, and explanation $\hat{z}_t$ should be semantically consistent with each other. For instance, if $\hat{z}_t$ states that the ith feature of x_t is irrelevant, the reward for $(x_t, \hat{z}_t, \hat{y}_t)$ should not change when the ith element of x_t changes. As an immediate result, $\hat{z}_t$ and $\hat{y}_t$ *must* be predicted jointly. The complex relationship between x_t, $\hat{z}_t$ and $\hat{y}_t$, makes learning to explain a structured prediction problem.
4. **Convergence.** The most important property of a learning algorithm is that, given enough examples, it outputs a model guaranteed to produce high-quality predictions. In our setting, this applies to the explanations produced by the model.

Solving all of the above challenges is far from trivial and beyond the reach of existing approaches, which focus on relevant features and do not guarantee convergence (**??**).

Method

We can now introduce EXPLEARN, a framework for learning self-explainable models based on contextual bandits that tackles the above challenges. In the following we focus on binary classification $\mathcal{Y}=\{0,1\}$, although extensions to multi-label settings are immediate. We also concentrate on *local* explanations, which target single instances, as opposed to global (model-wide) explanations (**?**).

The EXPLEARN Algorithm

To minimize the regret for the contextual bandit, we devise EXPLEARN, an algorithm that uses a principled learning strategy based on Gaussian Processes (GPs), a family of non-parametric models (**?**).

A GP is characterized by a mean function $\mu(x)$ and a covariance function (aka kernel) $k(x, x')$ that encodes prior knowledge about the second-order properties of the functions drawn from the GP[1]. GPs support closed-form Bayesian inference: conditioning a GP on evidence gives another GP whose mean and covariance functions can be derived analytically.

We now instantiate GPs on our setting. Let $\xi_t = (x_t, \hat{z}_t, \hat{y}_t)$ be the instance, explanation, and prediction handled by EXPLEARN in the tth iteration and $f_t = f(\xi_t) + \epsilon_t \in [0,1]$ be the corresponding user-supplied reward, where $\epsilon_t \sim \mathcal{N}(0, \sigma^2)$ are normally distributed independent noise variables. The *posterior GP* conditioned on the observed data $\Xi_T = \{\xi_1, \ldots, \xi_T\}$ and noisy rewards $\boldsymbol{f}_T = (f_1, \ldots, f_T)^\top$ has the following mean $\mu_T(\xi)$ and covariance $k_T(\xi, \xi')$:

$$\mu_T(\xi) = \boldsymbol{k}_T(\xi)^\top \Gamma_T \boldsymbol{f}_T \tag{3}$$

$$k_T(\xi, \xi') = k(\xi, \xi') - \boldsymbol{k}_T(\xi)^\top \Gamma_T \boldsymbol{k}_T(\xi') \tag{4}$$

where we use $\boldsymbol{k}_T(\xi) = (k(\xi_1, \xi), \ldots, k(\xi_T, \xi))^\top$, $K_T = [k(\xi, \xi') : \xi, \xi' \in \Xi_T]$, and $\Gamma_T = (K_t + \sigma^2 I)^{-1}$.

A kernel k on triples $\xi = (x, z, y)$ can be defined by composing base kernels over instances k_X, explanations k_Z, and labels k_Y (**?**). Two standard combination operators are the *direct sum* $\oplus$ and the *tensor product* $\otimes$. The direct sum of

[1]If no evidence is given, then $\mu(x)$ can be assumed without loss of generality to be 0. It is also customary to consider bounded variance functions, that is, $k(x, x) \leq 1$ for all x

Algorithm 1 The EXPLEARN algorithm

```
Input: Instances $\mathcal{X}$, Explanations $\mathcal{Z}$, Labels $\mathcal{Y}$, kernel over
    explanations $k$
1: for $t = 1, 2, \ldots, T$ do
2:     Receive instance $x_t$ from user
3:     Select explanation $\hat{z}_t$ and prediction $\hat{y}_t$ using Eq. 7
4:     Receive reward $f_t := f(x_t, \hat{z}_t, \hat{y}_t) + \epsilon_t$
```

two kernels $k : \mathcal{X} \to \mathbb{R}$ and $k' : \mathcal{X}' \to \mathbb{R}$ is a new kernel $(k \oplus k') : (\mathcal{X} \times \mathcal{X}') \to \mathbb{R}$:

$$(k \oplus k')((x_1, x_1'), (x_2, x_2')) = k(x_1, x_2) + k(x_1', x_2') \quad (5)$$

The tensor product is defined analogously:

$$(k \otimes k')((x_1, x_1'), (x_2, x_2')) = k(x_1, x_2) \cdot k(x_1', x_2') \quad (6)$$

GPs can provide a confidence on their own uncertainty. Hence, GPs are ideal for sampling regions in which the model is least certain about, with the purpose of increasing the confidence in that region. For this reason, GPs are well-suited for applications where querying the ground-truth reward function is expensive, like human-in-the-loop machine learning (**?**) or black-box Bayesian optimization (**?**). EXPLEARN exploits this property, too.

EXPLEARN estimates the reward function by assuming a GP prior on f and applies the CGP-UCB contextual bandit algorithm to obtain an accurate estimate of the reward function interactively (**?**). The pseudo-code is listed in Algorithm 1. CGP-UCB selects arms $(\hat{z}_t, \hat{y}_t)$ to be presented to the user by balancing between *exploitation* of arms with high mean reward and *exploration* of arms with uncertain (high variance) reward in a principled manner. In particular, an arm is chosen by maximizing

$$\hat{z}_t, \hat{y}_t = \max_{z,y} \mu_t(x_t, z, y) + \sqrt{\beta_t}\sigma_t(x_t, z, y) \quad (7)$$

The first term prefers an arm with high mean reward, while the second favors arms with high variance. The parameter β_t strikes a balance between exploration and exploitation and should slowly increase over time to favor more exploration. With an appropriate choice of β_t, CGP-UCB guarantees, under mild assumptions, no-regret as the number of iterations approach infinity:

Theorem 1 ((**?**)). *Let $\delta \in (0, 1)$, the norm $\|f\|_k$ be upper bounded by ω, and the noise variables ϵ_t form a martingale difference sequence (that is, $\mathbb{E}[\epsilon_t \mid \epsilon_1, \ldots, \epsilon_{t-1}] = 0$ for all $t \geq 1$) uniformly bounded by σ. Then, with probability at least $1 - \delta$, the cumulative regret of* CGP-UCB *is upper bounded by $\mathcal{O}^*(\sqrt{T\gamma_T\beta_T})$, where $\beta_t = 2\omega^2 + 300\gamma_t \ln^3(t/\delta)$ and γ_t depends only on the choice of kernel.*

Here, $\mathcal{O}^*$ denotes big-O notation with logarithmic factors suppressed and $\|f\|_k$ is the norm of the reward function f w.r.t. the chosen kernel k. In practice, convergence can often be improved by increasing β_t (**?**).

Notice that the technical assumptions occur often in practice. The martingale condition allows for correlated noise, which captures situations like annotation mistakes or temporary inattention. On the other hand, as long as the kernel is expressive enough the reward function has finite norm and CGP-UCB satisfies the no-regret property. Expressive kernels can be ensured through careful design, as discussed below.

Explanations and Kernels

EXPLEARN is natively supports different kinds of explanations. Here we discuss four representative forms of local explanations and their corresponding kernels.

Feature relevance A feature relevance explanation (FRE) z is a 0-1 vector whose jth element indicates whether the jth feature in x_t is relevant or not. For instance, a FRE for a diagnosis of pneumonia might highlight symptoms like "fever" and "dry cough" and exclude ones like "skin rash". This kind of explanations is commonly associated with, e.g., decision trees and rule sets (**??**).

In its simplest form, the space of possible FREs is the set of 0-1 vectors with as many elements as the instances x_t. Higher-dimensional feature spaces are also supported. Under the assumption that the user cares about how many (ir)relevant features were correctly identified by a FRE, the ground-truth reward will be proportional to the Hamming or Jaccard distance between a predicted FRE $\hat{z}$ and the ground truth z for x_t. A kernel that allows to encode this kind of reward is the set kernels.

Feature importance An explanation z is a real-valued vector in which the jth element represents the degree of agreement of jth towards the target decision. In the simplest case, elements are simply 0, 1, and -1. In our example, "fever" and "dry coughing" would be associated to a positive score and "skin rash" to a negative one. This kind of explanation is commonly associated to (discretized, sparse) linear models and neural nets (e.g., saliency maps).

In feature importance the user provides a score for the relevance of each feature. A ground-truth model outputs real valued explanations scaled for convenience in the $[0, 1]$ interval, where 1 indicates the maximum relevance. In this case the user cares of the overall score of each feature. As such, the ground-truth reward of the explanation is the cosine similarity $z^\top \hat{z}$ between the real and the predicted explanation. The choice of the kernel is more flexible, and depends on the data and the domain. Common choices for kernels are real-valued kernels, such as linear, polynomial, and RBF kernels.

Feature ranking An explanation z is a (sparse) preference relation that specifies the relative importance of different features. In this case, the importance value of each feature individually does not matter as long as the importance order is unaltered. This form of explanation is more interpretable for users as it eschews from checking the absolute value of each feature. In our example, "fever" could be ranked higher than "dyspnea". If the opposite was true, a different condition would be diagnosed, for instance common flu.

In feature ranking, the user provides a preference order for features. A ground-truth model outputs a natural number that corresponds to the position of each feature in the explanation vector. As the user is concerned only with the relative position of the features, the reward can be described as the amount of pairwise agreement among the ground-truth ranking z and the model's explanation ranking $\hat{z}$. The amount of agreement can be expressed as the (inverse) Kendall distance:

$$1 - \sum_{1 \le i,j \le n} \mathbb{1}(\pi(z(i)) < \pi(\hat{z}(j)) \wedge \pi'(z(i)) < \pi'(\hat{z}(j)))$$

where $\pi(z)$ maps feature values into positions and $\mathbb{1}$ is the indicator function. The Kendall distance is usually normalized between in the $[0, 1]$ interval, by dividing with the number of pairs $n(n-1)/2$. Similarly, the (inverse) Kendall distance is a kernel for explanations (**?**).

Feature traces An explanation z is the sequence of steps that leads to the classifier prediction for an instance x. In this case, the overall sequence is important in explaining the behaviour of a classifier. This kind of explanations are suitable for order-dependant predictor such as decision trees (DTs) or logic programs. A DT, for instance, justifies a diagnosis via a decision path from the root to the leaf. In the medical case, a DT could verify first that temperature is below 37, and if false verify whether it is above 39 and only then output "high-risk pneumonia". Traces allow a user to verify the algorithm steps in the prediction in a fine-grained manner.

In feature traces, the user provides a sequence of steps. A ground-truth model outputs a ordered subset of features that corresponds to a specific trace. The user in this case checks the sequence starting from the root and the model receives a reward proportional to the longest sub-sequence matching the ground-truth sequence. Suitable kernels for feature traces is the family of kernels on proof trees (**?**).

Discussion

Is learning to explain a sound learning task? If so, is EXPLEARN suited for it? In the following, we argue that both questions find an affirmative under reasonable assumptions.

Before proceeding, we start by assuming that the user's rewards are noisy but drawn from a fixed, ground-truth reward function $f : \mathcal{X} \times \mathcal{Z} \times \mathcal{Y} \to \mathbb{R}$. This assumption is quite general and underlies most human-in-the-loop applications of GPs (**??**).[2]

Theorem 1 states that CGP-UCB can achieve no-regret as long as a kernel k exists such that f has low norm in the corresponding reproducing kernel Hilbert space (RKHS). The question becomes whether such a kernel exists. One way to ensure that f has low norm w.r.t. k is to consider a *universal* kernel (**?**). Such kernels, which include the RBF kernel, guarantee that *all* functions, including f, can be approximated arbitrarily well in the corresponding RKHS[3]. This entails, on the one hand, that GPs equipped with a universal kernel can approximate any reward function f, and on the other, that EXPLEARN will eventually acquire a high-quality estimate of f.

Designing a universal kernel k over triples (x, z, y) is not trivial, especially if additional modeling assumptions have to be taken into account (**?**). A more straightforward alternative is to define universal kernels k_X, k_Z, and k_Y over instances, explanations, and labels and then compose them using a combination of tensor products and/or direct sums so that the resulting kernel k_{XZY} is also universal. It turns out that direct sum, however, is not viable in this sense. Indeed, any reward function $f_\oplus : \mathcal{X} \times \mathcal{X}' \to \mathbb{R}$ defined in the RKHS of $k \oplus k'$ can be written as:

$$\begin{aligned} f_\oplus(x, x') &= \textstyle\sum_{i=1}^n \alpha_i \cdot (k \oplus k')((x, x'), (x_i, x'_i)) \\ &= \textstyle\sum_{i=1}^n \alpha_i k(x, x_i) + \sum_{i=1}^n \alpha_i k'(x', x'_i)) \\ &= f(x) + f'(x') \end{aligned}$$

where $n \in \mathbb{N}, \alpha_1, \ldots, \alpha_n \in \mathbb{R}$, the x_i's belong to $\mathcal{X}$, the x'_i's to $\mathcal{X}'$, and f and f' are defined in the RKHS of the base kernels k and k', respectively. In other words, the direct sum imposes *additive independence* between the elements of $\mathcal{X}$ and the elements of $\mathcal{X}'$. Consider a kernel over triples (x, z, y) obtained using a direct sum, e.g., $k_{XZY} = k_X \oplus k_Z \oplus k_Y$. In this case, additive independence entails that the ranking of alternative explanations is independent of the instance and label. Indeed, any reward function $f_{XZY}(x, z, y)$ in the RKHS of k_{XZY} satisfies:

$$\begin{aligned} \forall x, y, z, z' \,.\, f_Z(z) \ge f_Z(z') \iff \\ f_{XZY}(x, z, y) \ge f_{XZY}(x, z', y) \end{aligned}$$

As a consequence, an explanation that is optimal for a given instance and label is still optimal for completely different instances and labels. This is not always desirable.

It follows that the direct sum does not preserve universality: even if k and k' are universal kernels, $k \oplus k'$ is not, as it cannot express reward functions that are not additively independent. In stark contrast, the tensor product does preserve universality: if k and k' are universal, so is $k \otimes k'$ (**???**). Hence, it is straightforward to build universal kernels $k = k_X \otimes k_Z \otimes k_Y$ that ensure with high probability that EXPLEARN attains no-regret. In practice, simpler, non-universal kernels can enjoy better sample complexity. A good trade-off between expressiveness and convergence could be achieved by leveraging automated kernel search, cf. (**?**) and follow-ups.

Advantages and Limitations

EXPLEARN has a number of important features. First, it is the first algorithm that sports learning guarantees for learning from explanations. Second, the models it learns are self-explainable and—unlike post-hoc explainers based on model distillation, like LIME (**?**)—they output exact explanations.

Many aspects of EXPLEARN can be improved. For instance, EXPLEARN relies on scoring feedback, which might not be very natural for end-users. This limitation can be

[2]In some cases (like concept or preference drift) users can be inconsistent and thus violate this assumption. This issue, however, is orthogonal to our contribution.

[3]Universality is usually defined for continuous data. A discrete counterpart is discussed in (**?**).

lifted by, e.g., asking for ranking feedback (?). More importantly, the space of explanations is combinatorial in nature. It follows that inferring a high-reward arm can be computationally non-trivial: depending on the kind of explanations and kernels used, Eq. 7 can become an **NP**-hard combinatorial optimization problem. Efficient solvers do exist for special cases, e.g., when the kernel is linear. A more general, yet efficient, alternative is to solve the arm selection problem by adapting recent techniques for combinatorial bandits (??). This is, however, non-trivial for general kernels. A simpler option is to solve Eq. 7 only approximately, leading to practical algorithms for more complex explanation types.

Experiments

We present preliminary evidence that EXPLEARN delivers on its promises. In particular, we evaluate whether EXPLEARN can **(Q1)** learn from feedback on both explanations and labels, **(Q2)** learn from and output different kinds of explanations, **(Q3)** outperform a random sampling strategy. To this end, we implemented EXPLEARN using Python 3 and scikit-learn and ran experiments on a 24 cores Intel(R) Xeon(R) CPU E5-2440 2.40GHz, 192GiB RAM. The experiments can be found at https://git.io/JL0Sv.

Kernels. For simplicity, we use RBF kernels for instances, explanations, and labels, and combine them either into a PROD kernel $k = k_X \otimes k_Z \otimes k_Y$ or a SUM kernel $k = (k_X \oplus k_Z) \otimes k_Y$, without any fine-tuning. As discussed above, fine-tuning could in principle attain better results.

Baseline and metrics. Besides the EXPLEARN algorithm, which uses the CGP-UCB arm selection strategy (as per Eq. 7), we implemented a "Random" baseline that asks for feedback on random arms. Performance is measured in terms of average cumulative regret, see Eq. 2. To ensure a fair comparison, the cumulative regret is computed relative to the best arm (that is, $\arg\max_{z,y} f(x_t, z, y)$) rather than the arm shown to the user $(\hat{z}_t, \hat{y}_t)$. Besides the random baseline, we test the PROD and SUM kernel. Although UCB generally performs better than Random, we believe our setup requires a more thorough investigation.

Datasets and Explanations

We evaluate EXPLEARN on two datasets with known ground-truth explanations and reward function. The ground-truth explanations are either synthesized or extracted from a pre-trained interpretable model.

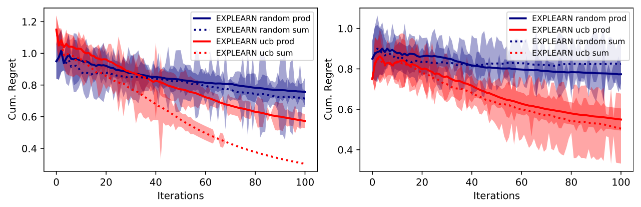
Colors (?) dataset consists of 5×5 images labeled as positive or negative according to one of two possible rules: rule 1 holds if the four corner pixels have the same color, while rule 2 holds if the three top-middle pixels have different colors. All images satisfy both rules or neither of them, making it impossible to disambiguate between the two rules based on labels alone. Both feature relevance and feature importance **ground-truth explanations** were generated, as follows. Feature relevance explanations are 0–1 vectors that identify those pixels relevant to the ground-truth label, namely the four corner pixels for rule 1 and the three top middle pixels for rule 2. Feature importance explanations are similar, except that the relevant pixels are weighted as $+1$ or -1 depending on whether they support or conflict with the label. Notice that feature importance explanations depend on the both image and the label. EXPLEARN was tested on both kinds of explanations. The models outputs feature relevance or feature importance explanations depending on the supervision. The **reward** is taken to be the cosine similarity between the ground-truth and the predicted explanation multiplied by 1 if the predicted and real label agree and by -1 otherwise.

Banknote (?) consists of 4-dimensional instances extracted from images of counterfeit and genuine banknotes. The data set includes 1372 instances, 610 counterfeit and 762 genuine. The **ground-truth explanation** for each instance is the path of a decision tree with depth between 5 and 10. The corresponding **model explanation** is any set of feature conditions in a root-to-leaf path. The **reward** is the Jaccard similarity among the ground-truth and the model explanation multiplied multiplied by 1 if the predicted and real label agree and -1 otherwise.

Results and Discussion

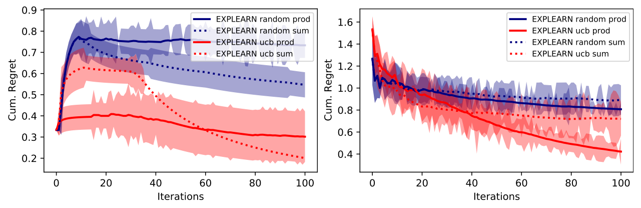
Figures 1–3 report the average and the standard deviation (in shaded color) of the cumulative regret as a function of the number of iterations. Recall that in each iteration the algorithm receives an instance, selects an explanation and a prediction and receives a reward. We report the results for the random sampling strategy and the UCB sampling strategy. We also report results for both the the sum and product kernel combination.

Q1: Does EXPLEARN learn from explanations? The first and foremost question is whether EXPLEARN can learn from feedback on explanations and labels. The plots show that the regret consistently decreases with the number of iterations / examples in all datasets, as expected. However, the results vary among different explanation types and datasets. In particular, for banknote (Figure 3) the regret falls fast in the first iterations but reaches a plateau around 60 iterations. The reason is probably the way explanations are constructed which disregard the order. **Q2: Does EXPLEARN work with different kinds of explanations?** To test the ability of EXPLEARN to learn from different explanation models we experiment with feature relevance (Figure 1a–2a), feature importance (Figure 1b–2b), and feature traces 3). In all cases EXPLEARN learns from explanations, although results are different. On the Colors dataset, with sparse features and explanations the sum kernel performs generally better, as the pixel in those images are not correlated. As such, the independence assumption is more realistic. On the Banknote dataset, all the variants perform similar, although the product kernel outperform the sum kernel. In the case of decision trees features are correlated. Therefore, the independence assumption of sum fails to capture such correlations. **Q3: Is UCB better than random?** Finally, we look at the question whether UCB is a better strategy than random. In general, the experiments confirm the theoretical advantage of UCB. On the other hand, a random exploration of new points might be beneficial in particularly challenging datasets a more complicated explanations, as in the case of Banknote in Figure 3. However, such a behaviour might be



(a) Feature relevance (b) Feature importance

Figure 1: Cumulative regret on Colors dataset on **rule 0**.



(a) Feature relevance (b) Feature importance

Figure 2: Cumulative regret on Colors dataset on **rule 1**.

determined by the lack of ordering in the reward model we applied in Banknote.

Related Work

Roughly speaking, explainable machine learning focuses on two major problems: designing white-box models that are inherently interpretable (cf. (**?**)) and understanding black-box models and their predictions (**??**). These forms of transparency are a prerequisite to identifying bugs and biases in learned models, but they are also insufficient to *correct* those models. In contrast, EXPLEARN explicitly imposes supervision on the explanations themselves.

Our work is strongly related to explanatory interactive learning (**???**) which integrates explanations into pool-based active learning (**?**) (with some exceptions (**?**)). In XIL, the machine picks instances and presents prediction and explanations for them to the user, who then supplies feedback on both the label and the explanation. This enables the machine to learn models that are "right for the right reasons" (**?**). EXPLEARN also relies on explanatory interaction, but it tackles sequential learning tasks in which instances arrive over time and are not chosen by the machine. In contrast to XIL, EXPLEARN also sports sound learning guarantees.

Most closely related to our work, self-explainable neural networks (SENNs) (**?**) natively output gradient-based explanations for their decisions and have been combined with explanatory active learning (**?**). The models learned by EXPLEARN are also self-explainable, but they are not restricted to gradient-based explanations. In addition, SENNs assume that explanations vary smoothly (in a Lipschitz sense), which is incorrect near the decision boundary. In contrast, EXPLEARN allows to more freely control how explanations generalize by choosing an appropriate kernel function.

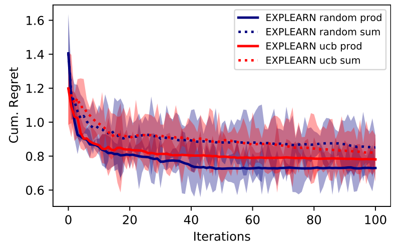


Figure 3: Banknote dataset, trace reward.

Conclusion

We introduced EXPLEARN, an approach for learning to explain by interacting with a human supervisor. EXPLEARN acquires self-explainable predictors that output explanations compatible with those supplied by their human supervisor. This is achieved by combining Gaussian Processes with a principled contextual bandit algorithm. Preliminary experiments on real-world and synthetic data sets indicate that, given enough examples, EXPLEARN successfully acquires models that output high-reward predictions and explanations. Of course, a more thorough evaluation of EXPLEARN is due. Another high-priority task is to speed up inference of the best arm by leveraging, e.g., knowledge compilation techniques used in combinatorial bandits (**??**). Other research directions involve strengthening the semantic consistency of instances-explanation-prediction triples by introducing specialized kernels, and enabling run-time customization of the kind and complexity of the explanations output by the model suitably to the user's needs, as done in (**?**).

Quas expedita hic fuga sequi quaerat eius doloribus commodi obcaecati, asperiores exercitationem nulla quasi tempora soluta impedit accusamus corrupti?Eius quidem minus facilis sapiente, voluptatibus quia vitae excepturi quod facere hic eligendi fuga, voluptate deleniti dignissimos, adipisci id soluta eum consectetur rem delectus?Saepe ut cumque ipsa explicabo sed iure quaerat repudiandae reprehenderit, dolorum eum hic qui deserunt temporibus itaque quam ea quibusdam?Debitis quae tempore deserunt placeat veniam a nesciunt, voluptatem porro dolor tenetur vitae provident exercitationem ut iure rerum, sint voluptates numquam velit molestiae saepe sunt temporibus accusantium dicta?Culpa ducimus dolores sed necessitatibus voluptate dignissimos accusamus esse, ipsum aliquid placeat dolorem harum corrupti mollitia quia quod consequuntur, suscipit sed facilis reiciendis veniam nesciunt, obcaecati reiciendis odit magni maxime dolore quaerat, sed inventore illo molestias excepturi voluptatibus.Fugit cum optio eum provident expedita necessitatibus iusto laborum, officia corrupti voluptatem, ut quia maxime unde laborum sed dolorem.Eius esse quod et ut, tempore quaerat nulla officia molestiae cupiditate sit quisquam corporis omnis?Odio excepturi ipsam quos sapiente voluptas eveniet, fuga quaerat minima illum iste accusantium eligendi quam nobis eum alias, cumque ratione aut eveniet voluptatem in eaque.Similique temporibus ipsum consequatur, voluptas perspiciatis nam inventore possimus iste eos magnam fuga est repellat debitis, beatae laborum ad non sapiente nemo dolores voluptates placeat officia quae eum, tempore culpa a provident adipisci quisquam.Cumque eum sed id, error modi quis est sequi ut quos maxime minima quibusdam?Ullam suscipit quos id facilis laudantium accusamus iusto