

Oblivious data for fairness with kernels

Arthur Katossky & Léo Houairi

Student	Time
Léo	8h15
Arthur	9h00

In “Oblivious data for fairness with kernels” (Grünewälder and Khaleghi 2021), Steffen Grünewälder and Azadeh Khaleghi consider the prediction of an outcome y from nonsensitive information x where there exists some sensitive information s that may be correlated with x . Dependence between x and s is a fairness issue if one wants to guarantee that predictions be independent from sensitive information $y \perp\!\!\!\perp s$. The authors restrict their discussion to kernel-based learning, that is the broad class of learning algorithms that, instead of relying directly on $(x_1 \dots x_n)$, may be expressed so as to rely on a $n \times n$ positive semi-definite matrix K_n summarizing the dependence between the observations.

In this context, they devise how to construct “oblivious” features $z = \phi(x)$ that both (a) guarantee the independence requirement $z \perp\!\!\!\perp s$ and (b) retain the information contained in x , in some maximal sense. After showing that a strict respect of both criteria is infeasible, they provide relaxed constraints and approximate solutions. The z features so constructed are to be used in the construction of the kernel matrix in place of x . Such features may be released and used for prediction, without sharing any confidential data.

The article being very technical and very thorough, we here propose a simplified version for the interested reader, with just enough to understand their application to binary classification (section 7.1). If you are familiar with (and not afraid by) [Reproducing Kernel Hilbert Spaces](#), [P-Donsker classes](#) or [Bochner-measurability](#), please refer to the original article.

Background

Imagine you have at our disposal data (X, S) where the X ’s are nonsensitive features but the S ’s are sensitive features. The goal of the article is to devise a procedure that allows to generate a new random variable Z that should ideally be

- Independent from S
- As close to X as possible, in some sense to be defined

We will see that later, the independence will be relaxed to a less binding constraint. This article particularly studies how to achieve this goal when we want to use kernel method. In this set-up, one basic idea could be to define Z as: $Z = X - E[X|S] + E[X]$ The difficulty being that the conditional expectation need to be estimated.

Mathematical set-up

Note

Je paraphrase pas mal le début de l'article ici.

We consider a probability space (Ω, \mathcal{A}, P) , a measurable space \mathcal{X} in which the random variable X takes its values and \mathcal{S} a measurable space \mathcal{S} in which S takes its values. Then, we consider \mathcal{H} the reproducing kernel hilbert spaces composed of functions $h : \mathcal{X} \rightarrow \mathbb{R}$, its feature map being $\phi(x) : \mathcal{X} \rightarrow \mathcal{H}$. Finally, we consider the \mathcal{L}^2 space of functions attaining values in \mathbb{H} (which is itself a space composed of functions).

Problem formulation and relaxations

At the begining, the problem considered is the construction of a random variable $Z : \Omega \rightarrow \mathcal{X}$ that is independant of S and closer to X than all other random variables respecting the independance criterion.

Then, the authors choose to do a first relaxation of the independance criterion. Instead of considering the independance as a criterion, they focus of the interactions of the random variables considered with functions (je paraphrase là). The independance criterion becomes that, $\forall h \in H, \quad \forall g \in L2$

$$E[h(Z) \times g(S)] = E[h(Z)] \times E[g(s)]$$

Because of the most important property of the RKHS, this condition can be rewritten, again $\forall h \in H, \quad \forall g \in L2$

$$E[\langle h, \phi(Z) \rangle \times g(S)] = E[\langle h, \phi(Z) \rangle] \times E[g(s)]$$

This is interesting, because $\phi(Z)$ leaves in the space \mathbb{H} but **does not cover all the space**. Indeed, if ϕ is continuous, $\phi(Z)$ is a low-dimensional manifold denoted \mathcal{M} thereafter. Then, the authors chose to introduce a new relaxation. Instead of considering $\phi(Z)$, they replace it by a random variable $\mathbf{Z} : \Omega \rightarrow \mathbb{H}$. So, we move from the “two-steps” where we had $Z : \Omega \rightarrow \mathcal{X}$ and then $\phi(Z) : \mathcal{X} \rightarrow \mathbb{H}$ to going directly from Ω to \mathbb{H} .

After this relaxation, the authors defined the notion of **H-Independence**. \mathbf{Z} and S are said H-independent iff $h \in \mathbb{H}$ and all bounded measurable $g : \mathbb{S} \rightarrow \mathbb{R}$, we have:

$$E[\langle h, \mathbf{Z} \rangle \times g(S)] = E[\langle h, \mathbf{Z} \rangle] \times E[g(s)]$$

This criterion is very close to the precedent one, except that again, we go directly to \mathbb{H} .

The problems becomes finding a \mathbf{Z} that is H-independent from S and as close as possible to $\phi(X)$ (in the $\|\mathbf{Z} - \phi(X)\|_2$, even if I am not sure to know what it means).

The crucial point is to remark that if \mathbf{Z} can be written as $\phi(W)$ for some W in \mathbb{X} , then it is easy to show that H-independence implies independence between a kernel estimator ($\langle \hat{h}, \mathbf{Z} \rangle$) and S . Of course, nothing guarantees that \mathbf{Z} lies in the image of ϕ but, if it is close to this image (ie close to the manifold \mathcal{M}), it could be **projected** on the manifold. At the bottom of page 6, the authors are defining a notion of distance of \mathbf{Z} to the manifold. Then, proposition 1 shows that $\exists W \in \mathbb{X}$ that attains this minimal value.

Section 4 is mostly about estimating empirically the distance from \mathbf{Z} to \mathcal{M} and showing how to bound the approximation.

In section 5, we are going back to the problem of choosing \mathbf{Z} . To do this, we recall the initial formula $Z = X - E[X|S] + E[X]$. But now we are working in the \mathcal{M} space, so it simply becomes:

$$Z = \phi(X) - E[\phi(X)|S] + E[\phi(X)]$$

“Such feature is H-independent from S . \mathbf{Z} is the best approximation of $\phi(X)$ in the MSE under H-independence.

Section 6 focuses on the generation of the oblivious features. The conditional estimation is estimated from the data using a plug-in estimator using half of the data (ie simply replacing probabilities by empirical probabilities). Subsection 6.2 focuses on how to control the error of this estimation. The other half of the data can then be used to create a predictor. Because the article focuses on kernel methods, the values of \mathbf{Z} never need to be computed, only their scalar products need be. For the end of the method, the article distinguishes between two cases:

- **Case 1 (M-Oblivious)**. This submethod works in cases where $Y|X$ is independent from S (which is sort of a markovian setting $S \rightarrow X \rightarrow Y$). In this case, a kernel is estimated from the second half of the data as usual, but the prediction on the new data point is done after transforming X into \mathbf{Z} .
- **Case 2 Oblivious**. When S can affect the label Y not only through X but also through other means, the authors propose to compute an oblivious kernel matrix, then to proceed as usual.

Concernant l'exemple

Note

Waste of computation with kernel os size $2n$ if we use just the upper left and bottom right corners?

The binary classification exemple in section 7.1 is pretty strange, and some of the claims (about parameter's values) are contradicted by the actual implementation. Note that the exemple falls in the **Oblivious** case. Indeed, the minority group is disadvantaged twice in the process:

- First, their grade X is diminished
- Secondly, their grade is compared to a higher threshold.

The authors insist on the difficulty of estimating how well a “fair” procedure works. Comparing the predictions to the real labels is not sufficient because this labels are considered, by definition, to be biased. In their synthetic case, the authors can compare their prediction with the true / fair label.

Application

The application uses synthetic data representing the grades of students discriminated according to some sensitive feature. $\tilde{s} \sim B(0.5)$ is the (binary) sensitive feature¹. The grades are biased towards the high-prestige status associated with $\tilde{s} = 1$, which the authors model as:

$$\tilde{x} \stackrel{def}{=} \tilde{x}_0 + \tilde{b}\mathbb{1}(\tilde{s} = 1) - \tilde{b}\mathbb{1}(\tilde{s} = 0)$$

... where $\tilde{x}_0 \sim \mathcal{N}_{[1,4]}(2.5, 0.5)$ is the baseline grade and $\tilde{b} \sim B(0.9)$ is the 1-unit bias for group $s = 1$ or against group $s = 0$ ².

Eventually, the authors construct a decision mimicking some decision made based on the students' official results. This decision is based on a mix of the true student abilities (as given by x_0) and the biased ones (as given by x). Namely, they set :

$$\tilde{y} = \mathbb{1}(\tilde{u} \geq \tilde{x}_0) \mathbb{1}(\tilde{x} + \tilde{s} \geq \theta)$$

¹In the code, half of the synthetic data has $s = 1$ the other half $s = 0$, deterministically:

```
unique_sensitive_feature_values=[0,1]
sensitive_features =
    [unique_sensitive_feature_values[0]] * n_samples +
    [unique_sensitive_feature_values[1]] * n_samples
```

²In the code:

... with $\tilde{u} \sim \mathcal{U}[0, 4]$ and θ a real parameter, arbitrarily set by the authors to be $\theta = \mathbb{E}\tilde{x}_0 = 2.5$ ³.

Contrary to the mathematical background, the implementation is not detailed in the paper, so we reconstruct it from the code. In [the SVM example](#), the substantial part is:

```
# X is training data (including sensitive data) of length 2n = 2*500
# S is X reduced to the sensitive data
# y_train is training predictions of length n
K = obl.build_K_lin(X,X)          # K has size 2n x 2n
O = obl.build_O_discrete(K,S)     # O has size n x n
clf = svm.SVC(kernel='precomputed')
clf.fit(O, y_train)
```

The `K = obl.build_K_lin(X,X)` line is straightforward, as it just builds the kernel matrix consisting in all dot products between line i and line j of X . On the other hand, the essence of the oblivious method proposed by the authors lay hidden behind the `O = obl.build_O_discrete(K,S)` line. It boils down to the following loop over the first $n = 500$

```
max_non_sensitive_feature_value = 4.0
min_non_sensitive_feature_value = 1.0
mu = 0.5 * (max_non_sensitive_feature_value+min_non_sensitive_feature_value)
Lower = generate_truncnorm_samples(
    n_samples, min_non_sensitive_feature_value,
    max_non_sensitive_feature_value, mu, sigma
)
Upper = generate_truncnorm_samples(
    n_samples, min_non_sensitive_feature_value,

    max_non_sensitive_feature_value, mu, sigma
)
non_sensitive_features =
    [Lower+stats.bernoulli(0.9).rvs(n_samples)*1]+
    [Upper+stats.bernoulli(0.9).rvs(n_samples)*1]
```

³The paper states that $\tilde{u} \sim \mathcal{U}[0, 1]$ but this contradicts the earlier statement that \tilde{x}_0 be truncated on $[1, 4]$. In the code, we see that $1(\tilde{u} \geq x_0) \sim \mathcal{B}(x_0/4)$, which means that in reality $\tilde{u} \sim \mathcal{U}[0, 4]$. Again, this ignores the fact that $\tilde{x}_0 \stackrel{a.s.}{>} 1$.

```
threshold = mu # mu is the mean of the truncated normal distribution from above
Y_Bernoulli_params = non_sensitive_features0 / max_non_sensitive_feature_value
Y = [stats.bernoulli(Y_Bernoulli_params[i]).rvs(1) for i in range(len(Y_Bernoulli_params))]

Y = Y * ((non_sensitive_features + sensitive_features >= threshold)*1)
```

observations, where all the averages `mean_iI`, `mean_IJ`, `mean_I`, `mean_i` and `Ephi` are computed on the second (independent) half. (This explains why `K` has size $2n \times 2n$ whereas `O` has size $n \times n$.)

```
for i in range(n): # n=500 (half the dataset length)
    for j in range(i,n):

        u = int(self.S_train[i]) # value of s for observation i
        v = int(self.S_train[j]) # value of s for observation j

        O[i,j] = K[i,j] # kernel value
        # 1. subtract individual-category average K[i, S=s]
        #     for s matching observed values
        - mean_iI[i,v] - mean_iI[j,u]
        # 2. add category-category average K[S=s1,S=s2]
        #     for s1, s2 matching observed values
        + mean_IJ[u,v]
        # 3. add individual average K[i,.]
        + mean_i[j] + mean_i[i]
        # 4. subtract category average K[S=s,.]
        #     with s matching observed values
        - mean_I[u] - mean_I[v]
        # 5. add general average K[.,.]
        + Ephi

        # save computation with symmetry
        O[j,i] = O[i,j]
```

Conclusions

[in our experiment], we are able to calculate the true (unbiased) errors as well. However, this is not always the case in practice. In fact, we argue that the question of how to evaluate fair classification performance is an important open problem which has yet to be addressed.

Source

Generating an oblivious random variable

Given a data-point (X, S) composed of non-sensitive and sensitive features X and S respectively, we can generate an oblivious random variable Z as

$$Z := \phi(X) - E_n^S \phi(X) + E_n(\phi(X)).$$

Most kernel methods work with the kernel matrix and do not need access to the features themselves. The same holds in our setting. More specifically, we never need to represent Z explicitly in the Hilbert space but only require inner-product calculations. In order to calculate the empirical estimates of the conditional expectation $E_n^S \phi(X)$ and of $E_n(\phi(X))$ in (9) we consider a simple approach whereby we split the training set into two subsets of size n , and use half the observations to obtain the empirical estimates of the expectations. The remaining n observations are used to obtain an oblivious predictor; we have two cases as follows.

Case 1 (M-Oblivious). The standard kernel matrix K is calculated with the remaining n observations and a kernel-method is applied to K to obtain a predictor g . When applying the predictor to a new unseen data-point (X, S) we first transform X into Z via (9) and calculate the prediction as $\langle g, Z \rangle$. As discussed in the Introduction, we conjecture that this approach is suitable in the case where the labels Y are conditionally independent of the sensitive features S given the non-sensitive features X , i.e. when S, X, Y form a Markov chain $S \rightarrow X \rightarrow Y$. As such we call this approach *M*-Oblivious.

Case 2 (Oblivious). Instead of calculating the kernel matrix K an oblivious kernel matrix, i.e.

$$\mathcal{O} = \begin{pmatrix} \|Z_1\|^2 & \cdots & \langle Z_1, Z_n \rangle \\ \vdots & \ddots & \vdots \\ \langle Z_n, Z_1 \rangle & \cdots & \|Z_n\|^2 \end{pmatrix}$$

is calculated by applying Equation (9) to the remaining training samples (X_i, S_i) before taking inner products. The oblivious matrix is then passed to the kernel-method to gain a predictor g . The matrix is positive semi-definite since $\mathbf{a}^\top \mathcal{O} \mathbf{a} = \left\| \sum_{i=1}^n a_i Z_i \right\|^2 \geq 0$, for any $\mathbf{a} \in \mathbb{R}^n$. The complexity to compute the matrix is $O(n^2)$ (see Appendix E for details on the algorithm). Prediction for a new unseen data-point (X, S) is now done in the same way as in Case 1.

Application

We carried out an experiment to mimic a scenario where a class of students should normally receive grades between 0 and 5, and anyone with a grade above a fixed

threshold $\theta = 2$ should pass. Half of the class, representing a “minority group”, are disadvantaged in that their grades are almost systematically reduced, while the other half receive a boost on average. More specifically, let the sensitive feature S be a $\{0, 1\}$ -valued Bernoulli random variable with parameter 0.5, and let X_0 be distributed according to a truncated normal distribution with support $[1, 4]$. Let the non-sensitive feature X , representing a student’s grade, be given by

$$X := (X_0 - B) \chi\{S = 0\} + (X_0 + B) \chi\{S = 1\}$$

where B is a Bernoulli random variable with parameter 0.9 independent of X_0 and of S . The label Y is defined as a noisy decision influenced by the student’s “original grade” X_0 prior to the S -based modification. More formally, let U be a random variable independent of X_0 and of S , and uniformly distributed on $[0, 1]$. Let $Y_0 := \chi\{U \geq X_0\}$ and define

$$Y := Y_0 \chi\{X + S \geq \theta\}.$$

Classification Error. In a typical classification problem, the labels Y depend on both X and S so when we remove the bias it is not clear what we should compare against when calculating the classification performance. Observe that our experimental construction here allows access to the true ground-truth labels

$$Y^* := \chi\{X_0 \geq \theta\}.$$

Therefore, we are able to calculate the true (unbiased) errors as well. However, this is not always the case in practice. In fact, we argue that the question of how to evaluate fair classification performance is an important open problem which has yet to be addressed.

Measure of Dependence. Let $\mathcal{F}_n := \sigma(X_1, \dots, X_n, S_1, \dots, S_n)$, $n \in \mathbb{N}$ be the σ -algebra generated by the training samples. In this experiment, we measure the dependence between the predicted labels \widehat{Y} produced by any algorithm and the sensitive features S as

$$\tilde{\beta}(\widehat{Y}, S) := \frac{1}{2} \sum_{s \in \{0, 1\}} \sum_{y \in \{0, 1\}} E \left| P(\widehat{Y} = y, S = s \mid \mathcal{F}_n) - P(\widehat{Y} = y \mid \mathcal{F}_n) P(S = s) \right|$$

which is closely related to the β -dependence (see, e.g. (Bradley, 2007, vol. I, p. 67)) between their respective σ -algebras. We obtain an empirical estimate of $\tilde{\beta}(\sigma(\widehat{Y}), \sigma(S))$ by simply replacing the probabilities in (14) with corresponding empirical frequencies.

Experimental results. We generated $n = 1000$ training and test samples as described above and the errors reported for each experiment are averaged over 10 repetitions. Figure 3 shows binary classification error vs. dependence between prediction and sensitive features for three different methods: classical Linear SVM,

Linear FERM, and Oblivious SVM. In Figure 3a the error is calculated with respect to the observed labels which are intrinsically biased and in Figure 3a the error is calculated with respect to the true fair classification rule Y^* given by (13). As can be seen in the plots, the true classification error of Oblivious SVM is smaller than that of the other two methods. Moreover, in both plots the β -dependence between the predicted labels produced by Oblivious SVM and the sensitive feature is close to 0 and is much smaller than that of the other two methods.

References

Grünewälder, Steffen, and Azadeh Khaleghi. 2021. “Oblivious Data for Fairness with Kernels.” *J. Mach. Learn. Res.* 22: 208–1.