Oblivious data for fairness with kernels

Arthur Katossky & Léo Houairi

Student	Time
Léo	9h00
Arthur	12h00

> **Note**
>
> léo: est-ce qu'on rajoute une petite note pour expliquer notre changement de notation par rapport à l'article (les x et z au lieu d'avoir des X et des Z partout). Et les tildes pour la partie sur la classification binaire, même si je ne comprends pas bien leur classification.

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 the sample $(x_1 ... x_n)$, may be expressed so as to rely only on a $n \times n$ positive semi-definite matrix K_n summarizing the dependence between the observations where $K_{ij} = k(x_i, x_j) = \langle \varphi(x_i), \varphi(x_j) \rangle$ with $k : \mathcal{H} \times \mathcal{H} \to \mathbb{R}$ is called the *kernel* and $\varphi : \mathbb{X} \to \mathcal{H}$ is the associated *feature map*. **Léo, k: c'est pas plutôt de** $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$

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. Such features may be released and used for prediction, without sharing any confidential data. 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.

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. The following report is organised in two sections, where we first introduce the theoretical background and then explore an application to binary classification with support vector machines.

Background

Imagine you have at our disposal data (x_n, s_n) where the x's are nonsensitive features but the S's are sensitive features. You want to provide a way to train a kernel-based model $f : \mathbb{Y} \to \mathbb{X}$ on (y_n, x_n) where $s \perp\!\!\!\perp \hat{y} \equiv f(x)$ but you want your method to be independent of the which specific model f you use. A wide-ranging solution is simply to replace x by an other variable z, keep x secret and perform the training on (y_n, z_n) instead. **Léo: ce serait pas plutôt $f : \mathbb{X} \to \mathbb{Y}$? J'ai un doute**

The goal of the article is thus to devise a procedure that allows to generate this new random variable z, that ideally should :

- be independent from s, a very hard constraint that the authors later relax
- be close to x, in some sense to be defined

The most straight-forward solution would be to define z as the residual of the orthogonal projection of x on s :

$$z \triangleq x - \mathbb{E}(x \mid s) + \mathbb{E}x$$

However, in the specific context of kernel methods, the training of a model does not require access to the individual x_is but rather only to the kernel matrix K, whose terms are $K_{ij} = k(x_i, x_j) = \langle \varphi(x_i), \varphi(x_j) \rangle$. So we might as well directly define $\mathbf{z}$ as the residual of the orthogonal projection of $\varphi(x)$ on s:

$$\mathbf{z} \triangleq \varphi(x) - \mathbb{E}[\varphi(x) \mid s] + \mathbb{E}\varphi(x) \tag{1}$$

… so that we get eventually $K_{ij} \simeq \langle \mathbf{z}_i, \mathbf{z}_j \rangle$ and $K \perp\!\!\!\perp (s_n)$.

The question are : why can't we guarantee strict independence? Do we loose anything when moving from z to $\mathbf{z}$? What relaxation of the strict independence constraint can we guarantee? What bounds do we get when we estimate the expectations in Equation 1 on specific samples ? Note that each of theses answers are complicated by the fact that φ (and thus $\mathbf{z}$) take values in a function space $\mathcal{H} \subseteq \mathbb{R}^{\mathbb{X}}$ that may not be finite.

> **i** Mathematical setup and notations
>
> We consider a probability space $(\Omega, \mathcal{A}, \mathbb{P})$ and a measurable space $(\mathbb{X}, \mathcal{X})$ (resp. $(\mathbb{S}, \mathcal{S})$) in which the random variables x and z (resp. s) take their values.
> For $a, b \in \mathbb{R}^d$, we note $a \cdot b = a^\top b$ the standard dot product.

Kernel methods

Des définition ici.
Then, we consider $\mathcal{H}$ the reproducing kernel hilbert spaces composed of functions $h : \mathbb{X} \to \mathbb{R}$, its feature map being $\phi(x) : X \to \mathcal{H}$. Finally, we consider the $\mathcal{L}^2$ space of functions attaining values in $\mathbb{H}$ (which is itself a space composed of functions).
Kernels and RKHSs considered in the articles are:

- The **linear kernel** $\forall (a, b) \in \mathbb{R}^d : k(a, b) = a \cdot b$. The feature map is $\varphi(x) = x$. The RKHS $\mathcal{H}$ is the set of all functions $h_{\mathbf{w}} : x \mapsto x \cdot \mathbf{w}$ for $\mathbf{w} \in \mathbb{R}^d$, endowed with the scalar product $\langle h_{\mathbf{u}}, h_{\mathbf{v}} \rangle = \mathbf{u} \cdot \mathbf{v}$. Noting that $f_{\mathbf{w}} = k(\sqcup, \mathbf{w})$ and that $f_{\mathbf{w}}(x) = k(x, \mathbf{w})$, we can verify the reproducing property: $\forall h \in \mathcal{H} : \langle h, k(\sqcup, x) \rangle = h(x)$.
- < The RKHS containing sin and $x \mapsto x^2$ the appears in fig.2, but I am not sure exactly how to define it. > (Léo: I am not sure it is necessary)

Independence is too strong a requirement

The ideal objective for oblivious features would be to construct a random variable z with values in $\mathbb{X}$ that is both independent from s and close to x. Assume distance is measured as the L^2 norm, we would define:

$$z = \arg \min_{a \in L^2(\mathcal{X})} \|a - x\|_2 \quad s.t. \quad a \perp\!\!\!\perp s$$

The authors rule out such a solution without much formalism :

> Completely removing the dependence of x on s without changing x drastically is an intricate task that is rife with difficulties.

We provide the following toy example. Consider the case where $x = f(s)$ for some measurable function $f : \mathbb{S} \to \mathbb{X}$. As $a \perp\!\!\!\perp s \iff \forall g$ measurable $: a \perp\!\!\!\perp g(s)$, this in particular true for $f(s) \equiv x$. Thus we have $z \perp\!\!\!\perp x$, which together with L^2 minimisation leads to $z \overset{a.s.}{=} \mathbb{E}x$. This defeats the purpose of preserving the variations of x in z.

Following the authors, we thus move from the strict independence assumption to the milder no-correlation assumption: $\mathbb{cov}(a_i, s_j) = 0$ for each component of a and s (assuming both $\mathbb{X}$ and $\mathbb{S}$ are finite-dimensional). If z and s were Gaussian, this would be an equivalence, but it is not true in the general case.

No-correlation between z and s is an excessive restriction

Decorrelation together with square-error minimisation is a well-known problem, whose solution is the residual of the projection of x onto s (where projection is defined from the vector product $\langle a, b\rangle = \mathbb{E}[ab]$) :

$$z = x - \mathbb{E}(x \mid s) + \mathbb{E}x$$

However there are at least two reasons for not liking this solution:

1. Covariance is a bilinear operator. This is nice because we can take advantage of linearity in computation. But the price to pay is that all non-linear dependencies will be missed. **[GIVE EXAMPLES]**
2. More fundamentally, we need the L^2 norm to have a sense on $\mathcal{X}$, which may not be the case for many objects on which kernel-methods are defined.

For both reasons, we may want to move away from defining z in the $\mathbb{X}$ space, and rather define $\mathbf{z}$ in the $\mathcal{H}$ space where the feature map $\varphi(x)$ lives. This helps on both sides as not only can covariance in an infinite feature space capture subtle forms of dependence that the linear case could not but we only have to define a distance on $\varphi(x)$, which is easy since $\varphi(x)$ is a $\mathbb{R}^d$-valued function (e.g. texts).

Relaxation

we should just replace z by phi(z) but instead, we use boldz. More flexibility as boldz need not have any corresponding preimage by phi. phi(calz) is a manyfold that is contained into the Z space.

Problem formulation and relaxations

At the begining, the problem considered is the construction of a random variable $Z : \Omega \to \mathbb{X}$ that is independant of S and closer to X than all other random variables respecting the independeance 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 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 $\mathbf{Z} : \Omega \to \mathbb{H}$. So, we move from the "two-steps" where we had $Z : \Omega \to \mathbb{X}$ and then $\phi(Z) : \mathbb{X} \to \mathbb{H}$ to going directly from Ω to $\mathbb{H}$.

After this relaxation, the authors defined the notion of $\mathbb{H}$**-Independence**. $\mathbf{Z}$ and S are said $\mathbb{H}$-independent iff $h \in \mathbb{H}$ and all bounded measurable $g : \mathbb{S} \to \mathbb{R}$, 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 $\mathbb{H}$-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 $\mathbb{H}$-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 chosing $\mathbf{Z}$. 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 $\mathbb{H}$-independent from S. $\mathbf{Z}$ is the best approximation of $\phi(X)$ in the MSE under $\mathbb{H}$-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 seting $S \to X \to Y$). 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 (denoted $\mathcal{O}$), then to proceed as usual.

Computing the oblivious matrix

The oblivious kernel matrix $\mathbb{O}$ is a n by n matrix containing the scalar products of the $\mathbf{z}$ ($\mathcal{O}_{ij} = \langle \mathbf{z}_i, \mathbf{z}_j \rangle$). Computing this matrix boils down to computing $\langle \mathbf{z}_i, \mathbf{z}_j \rangle \quad \forall i, j$. Using Equation 1, and given that we are doing estimations, we can write that:

$$\mathbf{z}_i = \varphi(x_i) - \mathbb{E}_n[\varphi(x_i) \mid s_i] + \mathbb{E}\varphi(x_i)$$

So, the scalar product that we want to estimate is:

$$\langle \varphi(x_i) - \mathbb{E}_n[\varphi(x_i) \mid s_i] + \mathbb{E}\varphi(x_i), \quad \varphi(x_j) - \mathbb{E}_n[\varphi(x_j) \mid s_j] + \mathbb{E}\varphi(x_j) \rangle$$

This scalar product can be dcomposed in 9 terms. In the article, the appendix E1 explains how each term can be boiled down to kernel evaluations, and provide a generic algorithm to do so (including a kernel $k(.,.)$ to be chosen).

Application to binary classification

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

In this section, we present an example of the application of the method to the binary classification. We refer both to the text of the article and to the code in their implementation [1]. Some of the claims made in the paper, especially about the value of parameters, are contradicted by the implementation. In such case, we will refer to what we found in the implementation.

[1] https://github.com/azalk/Oblivious

The application is based on synthetic data. In an imaginary scenario, students receive a grade and are suppose to vaidate if this grade exceed an arbitrarily chosen threshold θ. the students are discriminated according to some (binary) sensitive feature: $\tilde{s} \sim B(0.5)$ [2]. We refer to the students with $s = 1$ as the *minority group* and to the other students as the *advantaged group*.

The original grade is denoted $\tilde{x}_0$ and it follows a gaussian truncated between 1 and 4. Then, the grades are biased towards the adavantaged group, 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 $\tilde{b} \sim B(0.9)$, a Bernoulli random variable, is the 1-unit bias for group $s = 1$ or against group $s = 0$. So, most students in the adavantaged group will have $\tilde{x} > \tilde{x}_0$ and most students in the minority group will have $\tilde{x} < \tilde{x}_0$.

The decision to make a student validate (or not) is denoted by the letter y. The logical decision would be:

$$y^* \stackrel{\Delta}{=} \mathbb{1}(\tilde{x}_0 > \theta)$$

Instead, the decision is a mix of the student true ability (as given by x_0) and the biased one (as given by x):

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

with $\tilde{u} \sim \mathcal{U}(1, 4)$, or equivalently, $\mathbb{1}(\tilde{u} \geqslant \tilde{x}_0) \sim \mathcal{B}(\tilde{x}_0/4)$ **De mont point de vue il y a une erreur et ce devrait être** $\mathbb{1}(\tilde{u} \leqslant \tilde{x}_0)$**, ie plus ta note est élevé plus tu as de cahnce que** $\tilde{y}_0 = 1$

Observe that this exemple falls in the **oblivious case**, because, y given x is not independent from s. Indeed, the minority group is disadvantaged twice in the process: (i) their grade $\tilde{x}$ can be diminished, (ii) as compared to the other group, their grade is compared to a higher threshold $\theta - \tilde{s}$, because $\tilde{s} = 1$ for the advantaged group.

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 feature[^1]. The grades are biased towards the high-prestige status associated with $\tilde{s} = 1$, which the authors model as:

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

[2] In the implementation, half of the synthetic data has $s = 1$ and the other half $s = 0$, deterministically

… 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].

Sensitive feature in the code In 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
```

X0 in the code In the code:

```
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]
```

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 :

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

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

In the code: contradiciton regarding U The 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 $\mathbb{1}(\tilde{u} \geqslant x_0) \sim \mathcal{B}(x_0/4)$, which means that in reality $\tilde{u} \sim \mathcal{U}[0, 4]$. Again, this ignores the fact that $\tilde{x}_0 \overset{a.s.}{>} 1$.

```
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)
```

Note that the exemple falls in the **Oblivious** case, that is that $Y|X$ is not independent from S. Indeed, the minority group is disavantdged twice in the process:

- First, their grade $\tilde{x}$ is diminished.
- Secondly, their grade is compared to a higher threshold: $\theta - \tilde{s}$. Indeed, for the advantaged group, $\tilde{s} = 1$.

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 × 2n
O = obl.build_O_discrete(K,S) # O has size  n ×  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 $n = 500$ 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]
```

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. THe authors write:

> [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 (paper)

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 $S \to X \to Y$. 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 $O\left(n^2\right)$ (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

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

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 $\mathcal{F}_n := \sigma\left(X_1, \ldots, X_n, S_1, \ldots, S_n\right), n \in \mathbb{N}$ 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

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

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

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