Feature Selection using Stochastic Gates- Supplementary Material

August 12, 2020

Contents

1 Proof of Proposition 1

We now provide a proof for proposition 1 showing the equivalence between the stochastic optimization (7) and the deterministic one (6). We start by considering $\mathcal{S}'$ be a subset such that $\mathcal{S}^* \setminus \mathcal{S}' \neq \emptyset$. That is there exists some element in $\mathcal{S}^*$ that is not in $\mathcal{S}'$. For any such set $\mathcal{S}'$ we have that $I(\boldsymbol{X}_{\mathcal{S}'}; Y) < I(\boldsymbol{X}; Y)$. Indeed, if we let $i \in \mathcal{S}^* \cap \mathcal{S}'^c$ then we have

$$\begin{aligned} I(\boldsymbol{X}_{\mathcal{S}'}; Y) &\leq I(\boldsymbol{X}_{\setminus\{i\}}; Y) \\ &= I(\boldsymbol{X}; Y) - I(\boldsymbol{X}_i; Y | \boldsymbol{X}_{\setminus\{i\}}) \\ &< I(\boldsymbol{X}; Y), \end{aligned}$$

where the final inequality follows by Assumption 1. On the other hand, for any subset $\mathcal{S}'$ such that $S^* \subset S'$, based on Assumption 2 we get that $I(\boldsymbol{X}_{S'}; Y) = I(\boldsymbol{X}; Y)$. Now, when we consider the Bernoulli optimization problem we have

$$\max_{\boldsymbol{\pi}} I(\boldsymbol{X} \odot \tilde{\boldsymbol{S}}; \boldsymbol{Y}) \quad \text{s.t.} \quad \sum_l \boldsymbol{\pi}_l \leq k \text{ and } 0 \leq \boldsymbol{\pi}_l \leq 1.$$

The mutual information can be expanded as

$$I(\boldsymbol{X} \odot \tilde{\boldsymbol{S}}; Y) = \sum_{\tilde{s}} I(\boldsymbol{X} \odot \tilde{\boldsymbol{s}}; Y) p_{\boldsymbol{\pi}}(\tilde{\boldsymbol{S}} = \tilde{\boldsymbol{s}}),$$

where we have used the fact that $\tilde{\boldsymbol{S}}$ is independent of everything else. Our goal is to understand the form of the distributing of the random vector $\tilde{\boldsymbol{S}}$ that maximizes the objective subject to its constrains. Recall that in optimization (7) the coordinates of $\tilde{\boldsymbol{S}}$ are sampled at random, which is motivated by practical needs. This means that the distribution that is being optimized over $p_{\boldsymbol{\pi}}$ is a product distribution. Here, we will show that we can remove independence assumption. If we can show that the distribution found by solving the more general optimization problem is still a product distribution, then we obtain a solution to the original optimization (7).

Now, from above we know that the optimal value of the optimization is $I(\boldsymbol{X}_{\mathcal{S}'}; Y)$ for any set $\mathcal{S}^* \subset \mathcal{S}'$. Hence, any unconstrained distribution should place all of its mass on such subsets in order to maximize the mutual information. As a result $\sum_{l \in S^*} p_{\boldsymbol{\pi}}(\tilde{\boldsymbol{S}}_l = 1) = k$. However, there is an optimization constraint that $\mathbb{E}[\sum_l \tilde{\boldsymbol{S}}_l] \leq k$. Therefore, $\mathbb{E}[\tilde{\boldsymbol{S}}_l] = 0$ for any $l \notin \mathcal{S}^*$. Hence, the optimal solution is to select the distribution so that all of the mass is placed on the subset S^* and no mass elsewhere. As this is also a product distribution, this complete the proof of the claim.

2 Bridging the Two Perspectives

To motivate the introduction of randomness into the risk, we have looked at the feature selection problem from a MI perspective. Based on the MI objective, we have observed that introducing randomness into the constrained maximization procedure does not change the objective (See Proposition 1 in Section 4) Here we provide a relation between the MI objective (6) and the empirical risk (1), which supports our proposed procedure.

We first note that the MI maximization over the set $\mathcal{S}$ can be reformulated as the minimization of the conditional entropy $H(Y|\boldsymbol{X}_{\mathcal{S}})$ since $H(Y)$ does not depend on $\mathcal{S}$:

$$\max_{\mathcal{S}} I(\boldsymbol{X}_{\mathcal{S}}; Y) \iff \min_{\mathcal{S}} H(Y|\boldsymbol{X}_{\mathcal{S}}).$$

Recall that $\boldsymbol{X}_{\mathcal{S}} = \boldsymbol{X} \odot \tilde{\boldsymbol{S}}$. By Proposition 1, we rewrite the deterministic search over the set $\mathcal{S}$ by a search over the Bernoulli parameters $\boldsymbol{\pi}$:

$$\begin{aligned}\min_{\boldsymbol{\pi}} H(\boldsymbol{Y}|\boldsymbol{X} \odot \tilde{S}) &= \min_{\boldsymbol{\pi}} \mathbb{E}_{X,Y,\tilde{S}} \left[-\log P_{\boldsymbol{\theta}^*}(Y|\boldsymbol{X} \odot \tilde{\boldsymbol{S}}) \right] \\ &= \min_{\boldsymbol{\pi},\boldsymbol{\theta}} \mathbb{E}_{X,Y,\tilde{S}} \left[-\log P_{\boldsymbol{\theta}}(Y|\boldsymbol{X} \odot \tilde{\boldsymbol{S}}) \right],\end{aligned}$$

where the expectation is over $\boldsymbol{X}, Y \sim P_{\boldsymbol{\theta}_*}$, which is the true data distribution, and $\tilde{\boldsymbol{S}} \sim Bern(\tilde{\boldsymbol{S}}|\boldsymbol{\pi})$ and we put our model distribution as $P_{\boldsymbol{\theta}}$. Then we can rewrite the right hand side as:

$$\begin{aligned}\mathbb{E}_{X,Y,\tilde{S}} \log P_{\boldsymbol{\theta}^*}(Y|\boldsymbol{X} \odot \tilde{\boldsymbol{S}}) &= \mathbb{E}_{X,Y,\tilde{S}} \left[\log P_{\boldsymbol{\theta}^*}(Y|\boldsymbol{X} \odot \tilde{\boldsymbol{S}}) \frac{P_{\boldsymbol{\theta}}(Y|\boldsymbol{X} \odot \tilde{\boldsymbol{S}})}{P_{\boldsymbol{\theta}}(Y|\boldsymbol{X} \odot \tilde{\boldsymbol{S}})} \right] \\ &= \mathbb{E}_{X,Y,\tilde{S}} \left[\log \frac{P_{\boldsymbol{\theta}_*}(Y|\boldsymbol{X} \odot \tilde{\boldsymbol{S}})}{P_{\boldsymbol{\theta}}(Y|\boldsymbol{X} \odot \tilde{\boldsymbol{S}})} \right] + \mathbb{E}_{X,Y,\tilde{S}} \left[\log P_{\boldsymbol{\theta}}(Y|\boldsymbol{X} \odot \tilde{\boldsymbol{S}}) \right].\end{aligned}$$

Since $\mathrm{KL}(P_{\boldsymbol{\theta}^*}(Y|\boldsymbol{X} \odot \tilde{\boldsymbol{S}})||P_{\boldsymbol{\theta}}(Y|\boldsymbol{X} \odot \tilde{\boldsymbol{S}}))$ is non-negative, $\mathbb{E}_{\tilde{S}}\mathrm{KL}(P_{\boldsymbol{\theta}_*}(Y|\boldsymbol{X} \odot \tilde{\boldsymbol{S}})||P_{\boldsymbol{\theta}}(Y|\boldsymbol{X} \odot \tilde{\boldsymbol{S}}))$ is also non-negative because it is a weighted sum of non-negative terms. Noting that

$$\mathbb{E}_{\tilde{S}}\mathrm{KL}(P_{\boldsymbol{\theta}_*}(Y|\boldsymbol{X} \odot \tilde{\boldsymbol{S}})||P_{\boldsymbol{\theta}}(Y|\boldsymbol{X} \odot \tilde{\boldsymbol{S}}) = \mathbb{E}_{X,Y,\tilde{S}} \left[\log \frac{P_{\boldsymbol{\theta}_*}(Y|\boldsymbol{X} \odot \tilde{\boldsymbol{S}})}{P_{\boldsymbol{\theta}}(Y|\boldsymbol{X} \odot \tilde{\boldsymbol{S}})} \right]$$

we conclude that

$$\mathbb{E}_{X,Y,\tilde{S}} \left[-\log P_{\boldsymbol{\theta}^*}(Y|\boldsymbol{X} \odot \tilde{\boldsymbol{S}}) \right] \leq \mathbb{E}_{X,Y,\tilde{S}} \left[-\log P_{\boldsymbol{\theta}}(Y|\boldsymbol{X} \odot \tilde{\boldsymbol{S}}) \right].$$

If we consider the negative log likelihood of the target given the observations (i.e. $-\log P_{\boldsymbol{\theta}}(Y|\boldsymbol{X} \odot \tilde{\boldsymbol{S}})$) as a loss function L (in Eq. (1)), then we see that the minimizing the risk approximately maximizes the MI objective in (6).

3 Details of the Regularization Term

Here we provide a detailed description of our regularization term. For the vector of stochastic gates $\boldsymbol{z} \in \mathbb{R}^D$, the regularization term is expressed as follows:

$$\begin{aligned}\mathbb{E}_Z \left\| \boldsymbol{Z} \right\|_0 &= \sum_{d=1}^{D} \mathbb{P}[z_d > 0] = \sum_{d=1}^{D} \mathbb{P}[\mu_d + \sigma\epsilon_d > 0] \\ &= \sum_{d=1}^{D} \{1 - \mathbb{P}[\mu_d + \sigma\epsilon_d \leq 0]\} \\ &= \sum_{d=1}^{D} \{1 - \Phi(\frac{-\mu_d}{\sigma})\} \\ &= \sum_{d=1}^{D} \Phi\left(\frac{\mu_d}{\sigma}\right).\end{aligned}$$

Note that the derivative of the regularization term with respect to the distribution parameter μ_d is simply the Gaussian PDF:

$$\frac{\partial}{\partial \mu_d} \mathbb{E}_Z \left\| \boldsymbol{Z} \right\|_0 = \frac{\partial}{\partial \mu_d} \Phi(\frac{\mu_d}{\sigma}) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{\mu_d^2}{2\sigma^2}}.$$

We now turn our attention towards providing a scheme for selecting σ The effect of σ can be understood by looking at the value of $\frac{\partial}{\partial \mu_d} \mathbb{E}_Z ||\boldsymbol{Z}||_0$. In the first training step, μ_d is 0.5. Therefore, at initial training phase, $\frac{\partial}{\partial \mu_d} \mathbb{E}_Z ||\boldsymbol{Z}||_0$ is close to $\frac{1}{\sqrt{2\pi\sigma_d^2}} e^{-\frac{1}{8\sigma_d^2}}$. In order to remove irrelevant features, this term (multiplied by the regularization parameter λ) has to be greater than the derivative of the loss with respect to μ_d because otherwise μ_d is updated in the incorrect direction. To encourage such behavior, we set $\sigma = 0.5$, which is around the maximum of the gradient during the initial phase as shown in Fig. 1. Although the point that attains the maximum moves as μ changes, we empirically observe that setting $\sigma = 0.5$ performs well in our experiments when the regularization parameter λ is appropriately set. Note that in our implementation we divide the regularization term by the size of the features D. This rescaling normalizes the hyper-parameter search into a similar range.

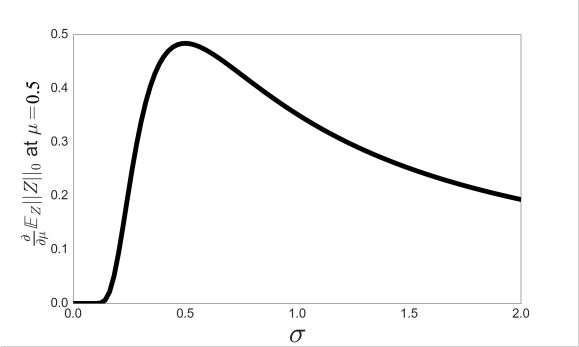


Figure 1: The value of $\frac{\partial}{\partial \mu} \mathbb{E}_Z ||\boldsymbol{Z}||_0|_{\mu=0.5} = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{8\sigma^2}}$ for $\sigma = [0.001, 2]$.

4 Issues in Gradient Estimation of Discrete Random Variables

In Section 3.1, we have introduced Bernoulli random variables $\tilde{s}_d, d = 1, ..., D$ with corresponding parameters π_d into the risk objective (4). Taking the expectation over the ℓ_0 norm of $\tilde{\boldsymbol{S}}$ boils down to the sum of the Bernoulli parameters π_d. However, a gradient-based optimization over the resulting objective suffers from high variance due to the discrete nature of $\tilde{\boldsymbol{S}}$. Here, we attempt to convey this problem by analyzing the risk term in the objective (4).

Using the Bernoulli paramterization the empirical risk $\hat{R}(\boldsymbol{\theta}, \boldsymbol{\pi})$ is expressed as

$$\sum_{\boldsymbol{z}:\{0,1\}^D} \left[\sum_{n=1}^{N} [L(f_\theta(\boldsymbol{z} \odot \boldsymbol{x}_n), \boldsymbol{y}_n] \prod_{d=1}^{D} \pi_d^{z_d} (1 - \pi_d)^{1 - z_d} \right].$$

In practice, as the outer sum involves enumerating 2^D possibilities of the indicator variables, one can replace the outer sum with Monte Carlo samples from the product of Bernoulli distributions $B(\boldsymbol{z}|\boldsymbol{\pi})$.

However, a Monte Carlo estimate of $\frac{\partial}{\partial \pi_d}\hat{R}(\boldsymbol{\theta}, \boldsymbol{\pi})$ suffers from high variance. To see this, consider the following exact gradient of the empirical risk with respect to π_d, which is

$$\sum_{\boldsymbol{z}:\{0,1\}^D, z_d=1} \left[L(\boldsymbol{z}) p_{z_{i\neq d}}\right] - \sum_{\boldsymbol{z}:\{0,1\}^D, z_d=0} \left[L(\boldsymbol{z}) p_{z_{i\neq d}}\right],$$

where $p(z_{i\neq d}) = \prod_{i\neq d}^{D} \pi_i^{z_i}(1-\pi_i)^{1-z_i}$, by absorbing the model $f_\theta(\cdot)$ and the data into $L(\cdot)$. Due to the discrete nature of $\boldsymbol{z}$, we see that even the sign of the gradient estimate becomes inaccurate if we can only access a small number of Monte Carlo samples. While a score-function estimator such as REINFORCE [Williams, 1992] can be used, it is known that the reparametrization trick is more effective for variance reduction [Jang et al., 2017, Maddison et al., 2016, Louizos et al., 2017].

5 Hard-Concrete Distribution (HC)

In the main text, we have compared the proposed embedded method using our STG distribution and an alternative based on the Hard-Concrete (HC). Here, we provide a full description for the HC distribution. The HC was introduced in [Louizos et al., 2017] as a modification of Binary Concrete [Jang et al., 2017, Maddison et al., 2016], whose sampling procedure is as follows:

$$\begin{aligned} u \sim U(0,1), L &= \log(U) - \log(1-U), \\ s &= \frac{1}{1+\exp(\frac{-(\log\alpha + L)}{\beta})}, \\ \bar{s} &= s(\zeta - \tau) + \tau, \\ z &= \min(1, \max(0, \bar{s})), \end{aligned}$$

where (τ, ζ) is an interval, with $\tau < 0$ and $\zeta > 1$. This induces a new distribution, whose support is $[0,1]$ instead of $(0,1)$. With $0 < \beta < 1$, the probability density concentrates its mass near the end points, since values larger than $\frac{1-\tau}{\zeta-\tau}$ are rounded to one, whereas values smaller than $\frac{-\tau}{\zeta-\tau}$ are rounded to zero.

The CDF of s is

$$Q_s(s|\beta, \log\alpha) = \text{Sigmoid}((\log s - \log(1-s))\beta - \log\alpha), \tag{1}$$

and so the CDF of $\bar{s}$ is

$$Q_{\bar{s}}(\bar{s}|\phi) = \text{Sigmoid}((\log(\frac{\bar{s}-\tau}{\zeta-\tau}) - \log(1-\frac{\bar{s}-\tau}{\zeta-\tau}))\beta - \log\alpha), \tag{2}$$

where $\phi = (\beta, \log\alpha, \zeta, \tau)$. Using this distribution to model the gates, the probability of a gate z being active is $1 - Q_{\bar{s}}(0|\phi)$ and can be written as

$$1 - Q_{\bar{s}}(0|\phi) = \text{Sigmoid}(\log\alpha - \beta\log\frac{-\tau}{\zeta}). \tag{3}$$

The hyperparameters β, ξ and τ are set as in [Louizos et al., 2017] and fixed throughout training, while α is learned to determine whether the gate is active or not. Note that L is distributed as a Logistic distribution, which causes high variance in $\bar{s}$ due to the heavy-tailness of the distribution. By replacing the logistic distribution with the Gaussian distribution, we reduce variance in gradient estimates, leading to stability of feature selection (See Section 8).

6 Additional Experiments

6.1 Details of Evaluating stochastic regularization schemes

The linear regression setting (which appears in Section 8) follows the analysis of [Wainwright, 2009]. Here we provide a full description of the experimental setting for this experiment as well as extended results.

Suppose the cardinality of the support $|\mathcal{S}| = k$ is known. Let $\boldsymbol{\beta}^* \in \mathbb{R}^D$ be a fixed sparse vector, such that $\beta_i^* \in \{-0.5, 0.5\}$ (with equal probability) if $i \in \mathcal{S}$, and $\beta_i^* = 0$ otherwise. Given a matrix of measurements $\boldsymbol{X} \in \mathbb{R}^{N \times D}$ with values drawn independently from $\mathcal{N}(0, 1)$, the response $\boldsymbol{y}$ is defined as

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta}^* + \boldsymbol{w}, \tag{4}$$

where the values of the noise $\boldsymbol{w}_i, i = 1, ..., N$ are drawn independently from $\mathcal{N}(0, 0.5)$.

Here, we reproduce this setting to evaluate the probability of perfect support recovery of $\boldsymbol{\beta}^*$, based on our method. As suggested by [Wainwright, 2009], we use a sparsity that scales with D such that $k = \lceil 0.4D^{0.75} \rceil$. For each number of samples N in the range $[10, 500]$, we run 200 simulations and count the portion of correctly recovered supports. We repeat this process for 2 different values of D and compare our performance to LASSO and the HC. For LASSO, the regularization parameter was set to its optimal value $\alpha_N = \sqrt{\frac{2\sigma^2 \log(D-k) \log(k)}{N}}$ [Wainwright, 2009]. For STG and HC we set $\lambda_N = C\alpha_N$, such that C is a constant, which is selected using a cross validated grid search in the range [0.1,10]. As evident from Fig. 2, even when restricting to linear functions our method has a clear advantage over LASSO. This implies that the ℓ_0 based penalty, although not convex in nature, allows us to recover the support of β^* using less samples. Furthermore, the proposed STG requires less samples than the HC distribution for perfect support recovery.

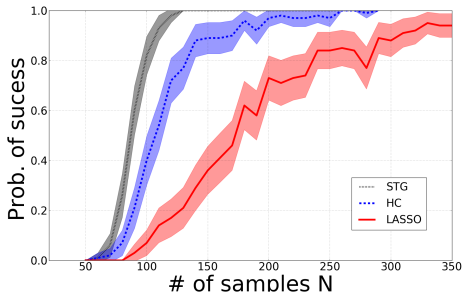


Figure 2: Probability of successfully recovering the support of a sparse signal $\boldsymbol{\beta}^* \in \mathbb{R}^D$ vs. the number of observations N. Comparison between the proposed method, HC and LASSO under a linear additive noise model $\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta}^* + \boldsymbol{w}$ using $D = 128$.

Furthermore, in Section 8 we have compared the STG to its deterministic counterpart. We define a deterministic gate using $\tilde{z}_d = g(\mu_d) = \max(0, \min(1, \mu_d))$, where μ_d is a parameter learned for each feature $d = 1, ..., D$.

In Fig. 3, we compare the values of the deteministic gates (DNC) and stochastic gates (STG) using $N = 60$. As depicted from this figure, the STG correctly identifies the active features (first 10) and sparsifies the non-active features. The deterministic version DNC does induce partial sparsity, but fails to remove some of the irrelevant features. Furthemore, it does not push the gates values of informative features up to 1.

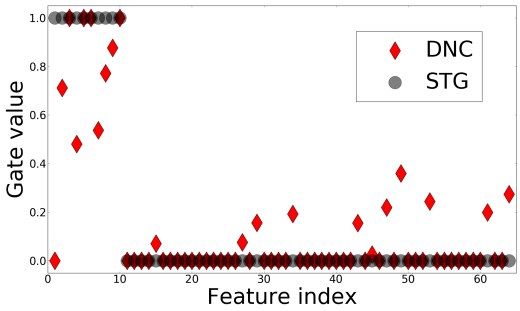


Figure 3: An example of the output gates, STG (z_d) and DNC ($\tilde{z}_d$) for $N = 60$. Each marker represents the gate value (y-axis) of the corresponding feature (x-axis), of which the first 10 features are informative. The deterministic gates are not pushed towards 1.

To evaluate the probability of support recovery using DNC, we apply additional thresholding to the gates. Specifically, we use a threshold of 0.5 to define if a feature is active or not. A comparison with STG, HC and LASSO in this setting appears in Fig. 4 (main text). The DNC is superior to LASSO, but inferior to the HC and STG. Which means that there is a true advantage of stochacity in our formulation.

6.2 Regression using synthetic and real datasets

In this section, we evaluate our method for regression tasks against several baselines (which are described in Section 6) and Sparse Random Fourier Features [Gregorová et al., 2018]. Following the same procedure as [Gregorová et al., 2018], the following functions are used to generate synthetic data (numbers next to the experiment code indicate total dimensions/relevant dimensions in the feature space): SE1: 100/5, $y = \sin(x_1 + x_3)^2 \sin(x_7 x_8 x_9) + \mathcal{N}(0, 0.1)$. SE2: 18/5, $y = \log((\sum_{s=11}^{15} x_s)^2) + \mathcal{N}(0, 0.1)$. SE3: 1000/10, $y = 10(z_1^2 + z_3^2)e^{-2(z_1^2+z_3^2)} + \mathcal{N}(0, 0.01)$, where each x_i is drawn from the standard Gaussian $\mathcal{N}(0, 1)$. For (SE3), the five consecutive coordinates are generated by $x_{5(j-1)+i} = z_j + \mathcal{N}(0, 0.1)$, where $z_j \sim \mathcal{N}(0, 1)$ for $i = 1, ..., 5$ and $j = 1, ..., 200$.

We also evaluate our method using three real datasets taken from the LIACC repository [1]: RCP: 21/- (computer systems activity), REL: 17/- (F16 elevators) and RAI: 39/- (F16 ailernos). For each synthetic dataset, we generate 30 different realizations, each consisting of 3000 samples. Then, we randomly split the data into training, validation, and testing sets using a $1 - 1 - 1$ ratio. The test root mean squared error averaged over the 30 runs is reported in Table 1. Our method outperforms all alternative methods for most of the datasets. Note that (SE1) is generated using a sine function; this favors the random Fourier feature-based method (SRFF).

Now, to demonstrate the cross validated grid search procedure we used for tuning λ, we focus on the RAI data as an example. We perform 5-fold cross validation using the STG and compute the MSE on the validation data for 100 values of λ logarithmically scaled in $[10^{-3}, 10^{1}]$. Here, we set this range such that for a fixed learning rate and number of epochs of 0.1 and 10,000 the minimum values of λ retain all features and the maximum value sparsifies all features. As demonstrated in Fig. 4, using a cross validation procedure for tuning λ provides a clear optimal value for this example (indicated as a red line).

[1] http://www.dcc.fc.up.pt/~ltorgo/Regression/DataSets.html

Table 1: Regression performance comparison in terms of root mean squared error. For the synthetic data (SE1-3), the mean and standard deviation are computed across 30 runs. For the real datasets, we use 30 different training-testing splits to evaluate the performance. The values for LASSO and SRFF are borrowed from [Gregorová et al., 2018]. DNN represents a deep neural network without feature selection.

Exp	Lasso	RF	SG-L1-NN	SRFF	DNN	HC	STG
SE1	0.29 (0.01)	0.30 (0.01)	0.29 (0.01)	**0.27** (0.01)	0.29 (0.01)	0.29 (0.01)	0.29 (0.01)
SE2	2.22 (0.10)	2.34 (0.17)	2.35 (0.18)	1.60 (0.10)	2.05 (0.11)	1.19 (0.31)	**0.87** (0.15)
SE3	0.68 (0.002)	0.50(0.01)	0.68 (0.01)	0.48 (0.03)	0.73 (0.02)	0.33 (0.05)	**0.14** (0.10)
RCP	9.69 (0.71)	3.52 (0.12)	9.64 (0.65)	2.52 (0.18)	2.89 (0.25)	2.74 (0.75)	**2.44** (0.08)
REL	0.47 (0.01)	0.58 (0.01)	0.44 (0.01)	0.31 (0.03)	0.61 (0.01)	0.33 (0.04)	**0.27** (0.002)
RAI	0.43 (0.02)	0.48 (0.003)	0.47(0.002)	0.41(0.02)	0.44 (0.01)	0.41 (0.01)	**0.39** (0.01)

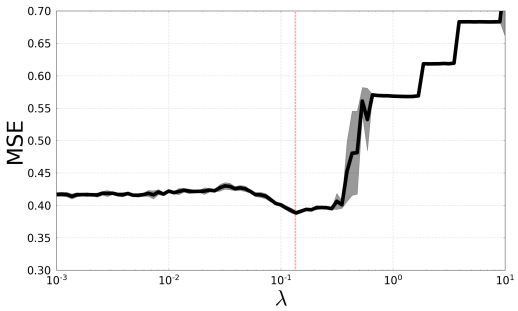


Figure 4: Demonstrating a cross validation procedure on the RAI dataset. We perform 5-folds cross validation and evaluate the MSE on the test set. The optimal $\lambda = 0.135$ which seems stable in this example based on the low standard deviation.

6.3 Noisy Binary XOR Classification

In the following evaluation, we consider the problem of learning a binary XOR function for a classification task. The first two coordinates, x_1 and x_2, are drawn from a "fair" Bernoulli distribution. The response variable is set as an XOR of the first two coordinates such that $y = x_1 \oplus x_2$. Coordinates $x_i, i = 3, ..., D$ are nuisance features also drawn from a binary, "fair" Bernoulli distribution. The number of points we generate is $N = 1,500$, of which 70 % are reserved for testing and 10% of the remaining training set are reserved for validation. We compare the proposed method to four embedded feature selection methods (LASSO [Tibshirani, 1996], C-support vectors (SVC) [Chang and Lin, 2011], deep feature selection (DFS) [Li et al., 2016], sparse-group regularized NN (SG-L1-NN) [Scardapane et al., 2017]). To provide more benchmarks, we also compare our embedded method against three wrapper methods (Extremely Randomized Trees (Tree) [Rastogi and Shim, 2000], Random Forests (RF) [Strobl et al., 2008] and Extreme Gradient Boosting (XGBOOST) [Chen and Guestrin, 2016]).

To evaluate the feature selection performance, we calculate the Informative Features Weight Ratio (IFWR). IFWR is defined as the sum of weights W_d over the informative features divided by the

sum over all weights. In the case of binary weights, the IFWR is in fact a recall measure for the relevant features.

The experiment is repeated 20 times for different values of D, and the average test classification accuracy and standard deviation are presented in Fig. 5(a) followed by the IFWR in Fig. 5(b). The number of selected features affects the accuracy; therefore, to treat all the methods in a fair manner, we tune the hyperparameter that controls the sparsity level using Optuna [Akiba et al., 2019], which optimizes the overall accuracy across different Ds. For instance, the wrapper methods (Tree, RF and XGBOOST) have a threshold value for feature retention. We retrain them using only those features whose weight is higher than the threshold. In terms of feature ranking (see Fig. 5(c)), only the tree-based and proposed methods (based on STG and HC) provide the optimal median rank of 1.5 for the two relevant features. Moreover, the ranking provided by STG is the most stable in comparison to all the alternative methods.

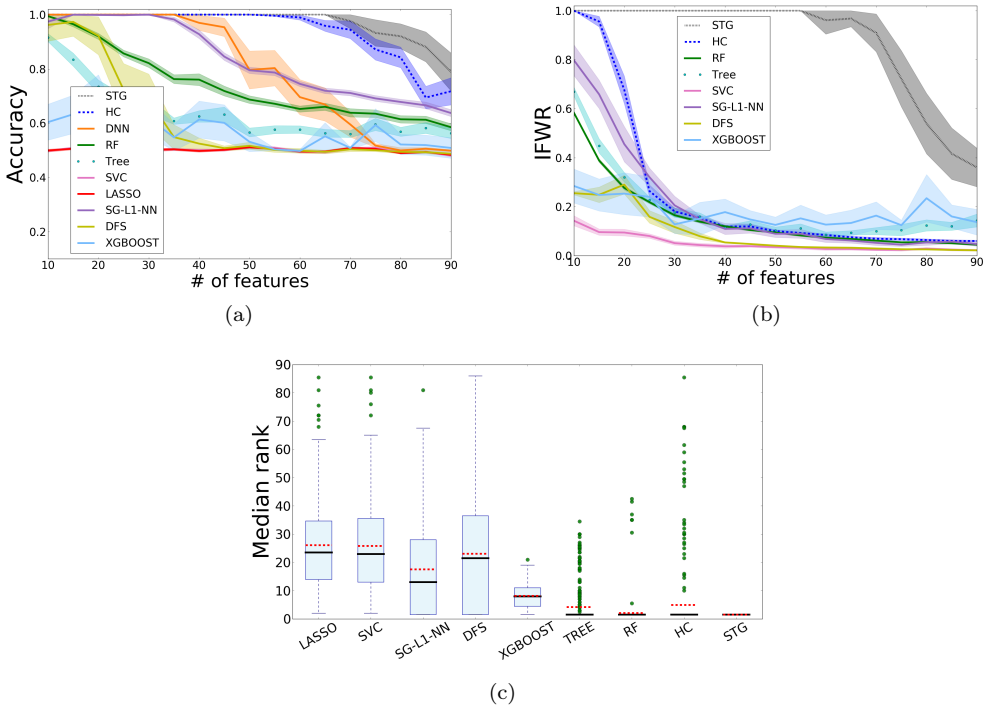


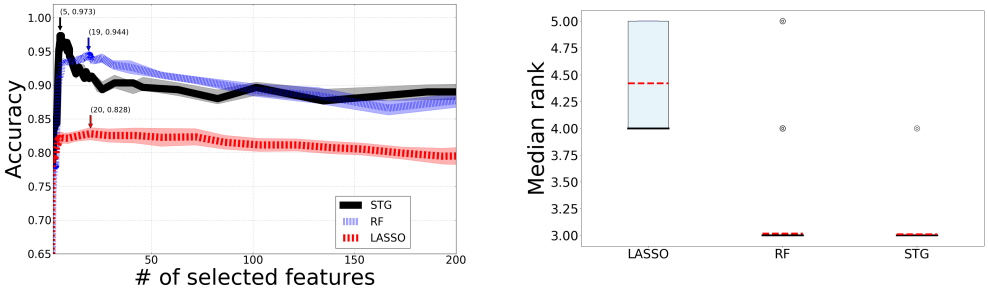
Figure 5: (a) Classification accuracy (mean and standard deviation) vs. the number of irrelevant noisy dimensions (D) for the XOR problem. (b) The Informative Features Weight Ratio (IFWR), lines are the means while the shaded regions represent the standard deviation. IFWR is the sum of weights W_d over the informative features divided by the sum over all weights. (c) Box plots for the median rank of the two informative features. The solid black and dashed red lines represent the median and mean of each method, respectively. The optimal median rank in this experiment is 1.5.

6.4 Identifying strongly relevant features using the MADELON dataset

To expand Section 8 of the main text, we further evaluate our proposed method in terms of its predictive power.

We vary the regularization parameter λ in the range $[0.01, 10]$ and compute the classification accuracy using 5 folds cross validation. In this example, we restrict our comparison to Random Forest and LASSO. We focus on Random Forest, as it was the strongest competitor to our method in all of our experiments; LASSO is evaluated because it is a widely-used embedded feature selection method.

As evidenced by Fig. 6(a), our method achieves the highest accuracy while using less features. Moreover, this figure depicts that peak performance occurs when selecting 5 features; thus, our method provides a clear indication of the true number of informative features. Both LASSO and RF, on the other hand, do not provide a clear indication of the true number of relevant features.

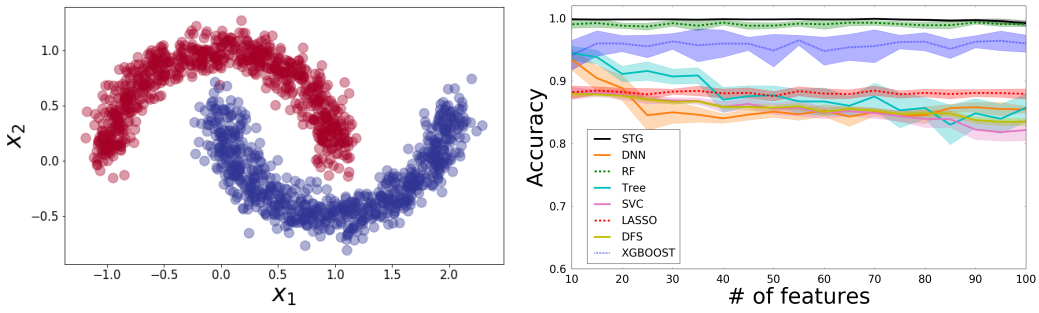


(a) (b)

Figure 6: (a) Classification accuracy on the MADELON datasets. We evaluate performance using 5-fold cross validation for different numbers of selected features. In this dataset, the first 5 coordinates are strongly relevant features, and the next 15 are weakly relevant. In that regime, the proposed method outperforms both RF and LASSO. (b) Box plots for the median rank of the 5 original informative features. The solid black and dashed red lines represent the median and mean of each method, respectively. The optimal median rank in this experiment is 3.

6.5 Two Moons classification with nuisance features

In this experiment, we construct a dataset based on "two moons" shape classes, concatenated with noisy features. The first two coordinates x_1, x_2 are generated by adding a Gaussian noise with zero mean and the variance of $\sigma_r^2 = 0.1$ onto two nested half circles, as presented in Fig. 7(a). Nuisance features $x_i, i = 3, ..., D$, are drawn from a Gaussian distribution with zero mean and variance of $\sigma_n^2 = 1$. We reserve the 70% as a test set, and use 10% of the remaining training set as a validation set. We follow the same hyperparameter tuning procedure as in the XOR experiment. The classification accuracy is in Fig. 7(b). Based on the classification accuracies, it is evident that for a small number of nuisance dimensions all methods correctly identify the most relevant features. The proposed method (STG) and Random Forest (RF) are the only methods that achieve near perfect classification accuracy for a wide range of nuisance dimensions. The other NN based method (DFS) seem to converge to sub-optimal solutions. We note that all the methods achieve the median rank 1.5, which is the optimal median rank in this example.



(a) (b)

Figure 7: (a) Realizations from the "Two moons" shaped binary classification class. x_1 and x_2 are the relevant features, $x_i, i = 3, ..., D$ are noisy features drawn from a Gaussian with zero mean and variance of 1. (b) Classification accuracy (mean and standard deviation based on 20 runs) vs. the number of irrelevant noisy dimension.

6.6 Convergence Comparison to Hard-Concrete distribution

In this section, we show additional experiments to compare STG with HC in terms of convergence speed.

The main difference between our proposed distribution (STG) and the Hard-Concrete [Louizos et al., 2017] distribution is that the latter is based on the logistic distribution, which has a heavier tail than the Gaussian distribution we have employed. As shown in Fig 8, the heavy-tailness results in instability during training. Furthermore, the STG converges much faster and more reliably than the feature selection method using the HC distribution on a the two-moons, XOR and MADELON datasets (see Subsection 6.4 and 6.5). A similar phenomenon is also demonstrated in the MNIST experiment (see Subsection 6.7).

6.7 Sensitivity Comparison to Hard-Concrete distribution using MNIST

Here we provide another axis to compare STG and HC by exploring how sensitive the performance and the sparsity level are to the regularization parameter λ.

As in Section 8, we attempt to distinguish between images of handwritten digits of 3's and 8's using samples from MNIST [LeCun et al., 1998]. The orientation and location of the digits is more or less the same throughout this dataset; therefore for these two classes (3's and 8's), we expect that some of the left side features (pixels) would be sufficient for the separation. Train and test splits are set based on the predefined MNIST separation $(50k, 10k)$. We use the MLP with ReLU activations and two hidden layers of size 300 and 100 [LeCun et al., 1998]. The STG extracts a relatively small number of selected features, which are positioned southwest and close to the center of the images. An example of 9 randomly selected samples overlaid with the weights of the selected features is presented in Fig. 9(a). Furthermore, we also evaluate the effect of λ on the the number of selected features and accuracy of the method.

We apply our method (STG) and its variant using the HC distribution to a randomly sampled training set of size $N = 1500$ and vary λ in the range of $[0.001, 0.01]$. In Fig. 9(b) we present the accuracy and sparsity level vs. the λ parameter. This experiment demonstrates the improved performance of the proposed distribution (STG) compared to the Hard-Concrete (HC [Louizos et al.,

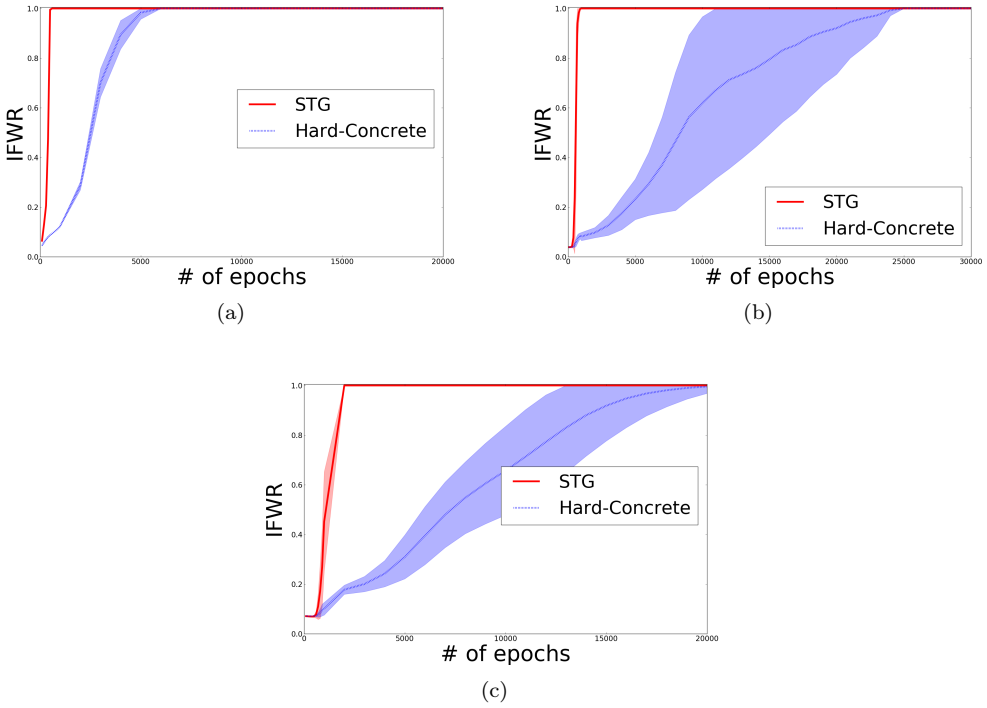


Figure 8: Comparison between STG and Hard-Concrete on the Two-Moon dataset (a), XOR (b) and MADELON (c). The shaded area represents the standard deviation, calculated by running the same experiment 10 times with different random initializations.

2017]), which was designed for neural net model compression. Not only that the overall accuracy is superior, but also it shows that the change in both accuracy and sparsity level is smoother with respect to λ, which suggests that the method is less sensitive to the choice of λ. Therefore using the STG may allow an easier regularization parameter tuning procedure compared with HC.

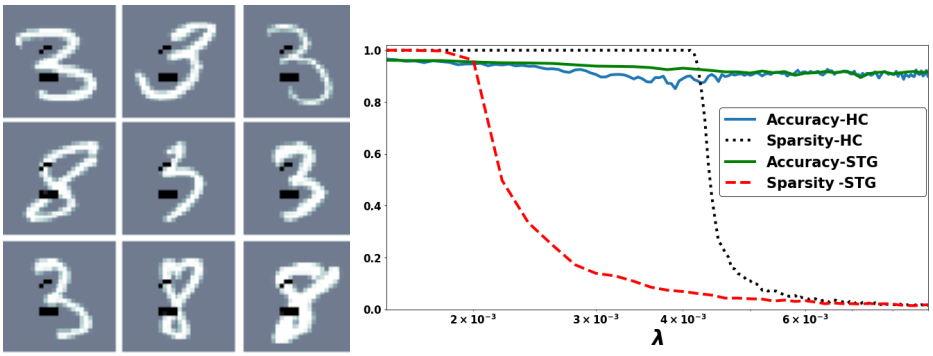
7 Reproducibility

Here we provide a full description of the procedures and datasets we have used in the experimental parts of the paper.

7.1 Datasets

7.2 Reuters Corpus Volume I

The Reuters Corpus Volume I (RCV1) consists of $800,000$ news stories manually labeled by 103 categories. This is a multilabel regime, where each story is assigned to multiple categories. Here we focus on a binary subset of this corpus, with $23,203$ stories, where we use $10\%, 8.5\%,$ and 81.5% for the train, validation and test set, respectively. The total number of feature is $47,236$. This example



(a) (b)

Figure 9: (a) Nine samples from MNIST (white) overlaid with the subset of 13 features (black) selected by STG. Based only on these features, the binary classification accuracy reaches 92.2%. For these nine randomly selected samples, all the 8's have values within the support of the selected features, whereas for the 3's there is no intersection. (b) The comparison of accuracy and sparsity level performance for λ in the range of $[10^{-3}, 10^{-2}]$.

demonstrates that our method is also effective in an extremely high dimensional regime of nonlinear function estimation.

7.3 Purified populations of peripheral blood monocytes (PBMCs)

Single-cell RNA sequencing (scRNA-seq) is a novel technology that measures gene expression levels of hundreds of thousands of individual cells. [Zheng et al., 2017], have subjected more than $90,000$ purified populations of peripheral blood monocytes (PBMCs) to scRNA-seq analysis. Here we focus on classifying two subpopulations of T-cells, namely the Naive and regulatory T-cells.

The data consists of $D = 11,382$ genes and $N = 20742$ cells, of which we only use 10% of the data for training. We apply the proposed method for different values of λ and report the number of selected features and classification accuracy on the test set. In this example, the STG identifies $20 - 30$ genes which are sufficient for the classification task.

7.4 Details of Cox Proportional Hazard Model

Survival times are assumed to follow a distribution, which is characterized by the survival function $S(t) = P(T > t)$. A hazard function, which measures the instantaneous rate of death, is defined by $h(t) = \lim_{\Delta t \to 0} \frac{P(t<T\leq t+\Delta t|T>t)}{\Delta t} = \frac{p(t)}{S(t)}$. We can relate the two functions in the following way: $S(t) = e^{-\int_0^t h(t)dt}$.

Proportional hazard models assume a multiplicative effect of the covariates x on the hazard function such that $h(t|\boldsymbol{x}) = h_0(t)e^{\boldsymbol{\theta}^T \boldsymbol{x}}$, where $h_0(t)$ is a baseline hazard function, which is often the exponential or Weibull distribution, and $\boldsymbol{\theta}$ is the parameter of interests.

One of the difficulties in estimating $\boldsymbol{\theta}$ in survival analysis is that a large portion of the available data is censored. However, in order to obtain estimates, Cox observed that it is sufficient to maximize

the partial-likelihood, which is defined as follows:

$$L(\boldsymbol{\theta}) = \prod_{T_i \text{uncensored}} \frac{e^{\boldsymbol{\theta}^T \boldsymbol{x}_i}}{\sum_{T_j \geq T_i} e^{\boldsymbol{\theta}^T \boldsymbol{x}_j}}.$$

In [Katzman et al., 2018], the authors propose DeepSurv, which uses a deep neural network model to replace the linear relations between the covariate $\boldsymbol{x}$ and $\boldsymbol{\theta}$, demonstrating improvements of survival time prediction over existing models such as CPH and the random survival forest [Ishwaran and Kogalur, 2007], [Ishwaran et al., 2008].

The Molecular Taxonomy of Breast Cancer International Consortium (METABRIC) dataset consists of gene expression data and clinical features for $1,980$ patients, and 57.72% have an observed death due to breast cancer with a median survival time of 116 months.

The METABRIC dataset involves $24,368$ features (genes). Most genes are irrelevant for outcome prediction. To demonstrate the advantage of STG in the context of survival analysis, we selected 16 well-known genes relevant for survival (out of the 24,368 genes) that correspond to the Oncotype DX test, a gene panel used for treatment decision making. We also include five additional clinical features (hormone treatment indicator, radiotherapy indicator, chemotherapy indicator, ER-positive indicator, and age at diagnosis). We then added 200 additional irrelevant gene variables that we selected randomly from the remaining list of genes.

After we omit the null values, the number of samples is 1969. We reserve the 20% for test, and use the 20% of the remaining training set as validation (that is, train: 1260, valid: 315, test: 394 samples).

Experimental Detail For DeepSurv, we manually select the architecture using the validation set so that we obtain a similar performance reported in [Katzman et al., 2018]. The learning rate decay is set to 1. The learning rate and the regularization parameter are optimized via Optuna Akiba et al. [2019] using the validation set, where the search range is set$[1e-3, 1]$ for the learning rate and $[1e-3, 1]$ for λ. The hyperparameters used in the experiment are the following: architecture :[60, 20, 3], activation: Selu (as suggested by [Katzman et al., 2018]), learning rate : 0.152, learning rate decay: 1.0, σ : 0.5, λ : 0.023, training epoch: 2000. Note that to see the effect of feature selection, we used the hyperparameters optimized for DeepSurv to test our method (Cox-STG).

7.5 Implementation details

Datasets are first split into train, validation and test. Validation is always 10% of the train, while the exact ratios between train and test is detailed for each experiment separately (see Table 1). All the neural network weights are initialized by drawing from $\mathcal{N}(0, 0.1)$ and bias terms are set to 0 (following Xavier initialization Glorot and Bengio [2010]). We use SGD for all the experiments, except for the Cox model where we use Adam. All the experiments are conducted using Intel(R) Xeon(R) CPU E5-2620 v3 @2.4Ghz x2 (12 cores total). We set the number of Monte Carlo samples $K = 1$, which worked well in our experiments. Hyper-parameters for all method are tuned using Optuna [Akiba et al., 2019]. Optuna is a hyper-parameter optimization software that supports pruning and parallel computing across GPUs. We run $n = 1000$ trails with parameters search ranges as described in Table 2.

For linear regression experiment, we use 0.1 as a learning rate. For the XOR problem, the exact architectures used for the NN based methods are: (STG/HC): $[476, 490, 14]$ with Tanh, (DFS): $[100, 10]$ with Tanh, (SG-NN): $[100, 10, 5]$ with Tanh. For the two moons we use (STG): $[490, 406, 18]$ with Tanh, (DFS): $[158, 27, 224]$ with Tanh, (SG-NN): $[88, 28, 27]$ with Tanh. For the XOR problem,

Table 2: List of the search range for the hyperparameters used in our expirements

Param	Search range
# dense layers	[1,5]
# hidden units	[10, 500]
activation	[Tanh, Relu, Sigmoid]
LR	[1e-4, 1e-1]
n-epoch (DFS, SG-NN)	[50, 20000]
α (SG-NN)	[1e-3, 10]
λ (SG-NN)	[1e-7, 10]
λ (STG, DFS)	[1e-3, 10]
λ (LASSO)	[1e-5, 1]
n-est (RF, XGB, ERT)	[5,100]
n-boost-round (XGB)	[1,100]
Thresh (RF, XGB, ERT)	[1e-7,1]
max-depth (XGB)	[1,5]
c (SVC)	[1e-7, 1]

we attempted to use Optuna to optimize parameters of DFS and SG-L1-NN, but we ended up using the architecture suggested by the authors [Li et al., 2016, Scardapane et al., 2017] as they outperform the values suggested by Optuna. The number of epochs used for the XOR problem is $20k, 14k, 800$ for STG/HC, DFS and SG-L1-NN respectively. Regularization parameters are $0.17, 3.3e-5$ and $3e-5$ respectively. The number of epochs used for the two-moons problem is $20K, 1570, 708$ for STG/HC, DFS and SG-NN respectively. Regularization parameters are $0.48, 9e-3$ and $1e-3$ respectively. We note that the regularization parameters and learning procedure is different in nature, as we use an ℓ_0 type penalty. For the PBMC experiment, the architecture use us $[27, 10, 383]$ with Tanh activations, a learning rate was 0.0036, batch size is 1000 and the number of epochs 8000. The hyperparameter λ varies in the range $[0.001, 0.11]$ to achieve different levels of sparsity. For MADELON, GISETTE, ISOLET, we use the architecture optimized for the binary XOR classification. The number of epochs used are $20K, 20K, 4k$, the learning rate are $0.06, 0.2, 0.1$, batch size are $200, 1000, 40$. The parameters used are the following: (SE1) architecture $[600, 200, 100, 50]$ with ReLu activations, number of epochs is 5, $\lambda : 5$, learning rate 0.0001 (SE2) architecture $[600, 300, 150, 60, 20]$ with ReLu activations, number of epochs is 2000, $\lambda : 5$, learning rate is 0.001 (SE3) architecture $[600, 300, 150, 60, 20]$ with ReLu activations, num epochs : 1000, $\lambda : 1$, learning rate 0.005. (RCP) architecture $[1000, 300, 150, 60, 20]$ with ReLu activation, num-epochs: 2000, $\lambda : 5.0$, learning rate 0.001. (REL) architecture $[26, 91, 63]$ with ReLu activation, number of epochs: 1600, $\lambda : 0.031$, learning rate 0.007. (RAI) architecture $[10, 177]$ with ReLu activation, number of epochs: 1800, $\lambda : 0.019$, learning rate is 0.002. Architectures for SE1-SE3 and RCP where tuned manually. The ratio of train/test/valid split is 1:1:1 for synthetic regression data. For the real regression data (RCP and REL), the train size is 6000, the test and valid size is 1000 samples. For RAI the train size is 5000, the test and valid size is 1000 samples. For the Friedman data we use Tanh with an architecture of $[500, 200, 100, 20]$ (HC/STG/DFS/SG-NN) with Tanh activations, a learning rate of 0.2 and a batch size of 200. For BASEHOCK (STG/HC) the architechture used is $[500, 105, 25]$ with Tanh activations, a batch size of 50, learning rate of 0.5 and 2000 epochs. For RELATHE (STG/HC) the architechture used is $[500, 200, 10]$ with Tanh activations, a batch size of 40, learning rate of 0.008 and $10k$ epochs. For COIL20 (STG/HC) the architechture used is $[32, 438, 158, 20]$ with Tanh activations, a batch size of 150, learning rate of

0.12 and $10k$ epochs. For PCMAC (STG/HC) the architechture used is $[109, 25, 455]$ with Tanh activations, a batch size of 450, learning rate of 0.5 and 6000 epochs. For RCV1 (STG/HC) the architechture used is $[500, 100]$ with Tanh activations, a batch size of 500, learning rate of 0.5 and 650 epochs. For MNIST (STG/HC) the architechture used is $[300, 100]$ with Relu activations (as in LeCun et al. [1998]), a batch size of 200, learning rate of 0.1 and 250 epochs.

In any experiment where we count the number of active features, we evaluate the set of indices such that: $\{d : \min(1, \max(0, \mu_d)) > 0\}$ after training. In order to define the IFWR, for STG, the d^{th} feature weight is set to $\max(0, \min(1, \mu_d))$. For other neural net based methods, it is given by $\sum_j W_{dj}$, where W is the weight matrix of the first layer. For other methods we just used the feature relevance returned by the trained model. Finally, the LASSO's IFWR in the XOR experiment was omitted from the manuscript as it suffered from high variance.

Regarding the comparison performed in the two-moons and XOR problem, we believe that adding IFWR along with classification accuracy versus number of feature selected provides a complementary perspective in demonstrating the efficacy of feature selection techniques. We emphasize that our goal is not to just rank features but select features by assigning the weight of 0 to irrelevant features while simultaneously obtaining good predictive accuracy.

References

T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama. Optuna: A next-generation hyperparameter optimization framework. *KDD*, 2019.

C.-C. Chang and C.-J. Lin. Libsvm: A library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):27, 2011.

T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794. ACM, 2016.

X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

M. Gregorová, J. Ramapuram, A. Kalousis, and S. Marchand-Maillet. Large-scale nonlinear variable selection via kernel random features. *arXiv preprint arXiv:1804.07169*, 2018.

H. Ishwaran and U. B. Kogalur. Random survival forests for r, 2007.

H. Ishwaran, U. Kogalur, E. Blackstone, and M. Lauer. Random survival forests. *Annals of Applied Statistics*, 2(3):841–860, 9 2008. ISSN 1932-6157. doi: 10.1214/08-AOAS169.

E. Jang, S. Gu, and B. Poole. Categorical reparameterization with gumbel-softmax. 2017. URL https://arxiv.org/abs/1611.01144.

J. L. Katzman, U. Shaham, A. Cloninger, J. Bates, T. Jiang, and Y. Kluger. Deepsurv: personalized treatment recommender system using a cox proportional hazards deep neural network. *BMC Medical Research Methodology*, 18, 2018.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Y. Li, C.-Y. Chen, and W. W. Wasserman. Deep feature selection: theory and application to identify enhancers and promoters. *Journal of Computational Biology*, 23(5):322–336, 2016.

C. Louizos, M. Welling, and D. P. Kingma. Learning sparse neural networks through l0 regularization. *CoRR*, abs/1712.01312, 2017.

C. J. Maddison, A. Mnih, and Y. W. Teh. The concrete distribution: A continuous relaxation of discrete random variables. *CoRR*, abs/1611.00712, 2016. URL http://arxiv.org/abs/1611.00712.

R. Rastogi and K. Shim. Public: A decision tree classifier that integrates building and pruning. *Data Mining and Knowledge Discovery*, 4(4):315–344, 2000.

S. Scardapane, D. Comminiello, A. Hussain, and A. Uncini. Group sparse regularization for deep neural networks. *Neurocomput.*, 241(C):81–89, June 2017. ISSN 0925-2312. doi: 10.1016/j.neucom.2017.02.029. URL https://doi.org/10.1016/j.neucom.2017.02.029.

C. Strobl, A.-L. Boulesteix, T. Kneib, T. Augustin, and A. Zeileis. Conditional variable importance for random forests. *BMC bioinformatics*, 9(1):307, 2008.

R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

M. J. Wainwright. Sharp thresholds for high-dimensional and noisy sparsity recovery using ℓ_1-constrained quadratic programming (lasso). *IEEE Transactions on Information Theory*, 55(5):2183–2202, May 2009. ISSN 0018-9448. doi: 10.1109/tit.2009.2016018. URL http://dx.doi.org/10.1109/tit.2009.2016018.

R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.

G. X. Zheng, J. M. Terry, P. Belgrader, P. Ryvkin, Z. W. Bent, R. Wilson, S. B. Ziraldo, T. D. Wheeler, G. P. McDermott, J. Zhu, et al. Massively parallel digital transcriptional profiling of single cells. *Nature communications*, 8:14049, 2017.