SToG: Python Library for Feature Selection in Neural Networks

Altay Eynullayev Sergey Firsov Gleb Karpeev Denis Rubtsov

Motivation

Feature selection is a fundamental challenge in machine learning, particularly when dealing with high-dimensional data where the number of features can greatly exceed the number of samples. Traditional feature selection methods such as LASSO work well for linear models but struggle to capture nonlinear relationships inherent in complex datasets. Deep neural networks, while powerful, often suffer from overfitting and computational inefficiency when processing datasets with thousands or millions of features.

SToG (Stochastic Gating) is a Python library that addresses these challenges by implementing four state-of-the-art deep learning-based feature selection methods. The library enables end-to-end differentiable feature selection through stochastic gating mechanisms, allowing neural networks to simultaneously learn predictive models while identifying the most relevant features. This approach is particularly valuable in domains such as genomics, medical diagnostics, finance, and any field where interpretability and computational efficiency are critical.

The motivation behind SToG is threefold: (1) to provide researchers and practitioners with robust tools for nonlinear feature selection, (2) to enable feature selection with limited labeled data through self-supervised learning, and (3) to support both static feature pruning and dynamic conditional computation for improved inference efficiency.

1. Introduction

1.1 Background and Motivation

Feature selection aims to identify a small subset of relevant features from a potentially very large feature space. This task is crucial for several reasons. Firstly, fewer features mean faster training and inference. Secondly, removing irrelevant features reduces overfitting. Thirdly, simpler models are easier to understand and explain and in domains like genomics, measuring fewer biomarkers reduces expenses.

Traditional approaches to feature selection fall into three categories: *filter methods*, aiming at feature selection based on statistical measures (e.g., correlation, mutual information) independent of the learning algorithm, *wrapper methods*, evaluating feature subsets by training a model and measuring performance (e.g., recursive feature elimination) and *embedded methods*, performing feature selection as part of the model training process (e.g., LASSO, Elastic Net)

While LASSO and related methods excel at linear feature selection, they are limited to linear models and suffer from weight shrinkage. Modern deep learning applications require feature selection methods that can: *a)* handle nonlinear relationships between features and targets, *b)* work effectively with high-dimensional, low-sample-size datasets, *c)* provide end-to-end differentiable training pipelines, *d)* account for feature correlations and redundancy.

SToG addresses these requirements by implementing four complementary approaches based on stochastic gating mechanisms.

1.2 SToG Library Overview

SToG is an open-source Python library that provides implementations of four cutting-edge feature selection methods:

1. **STG (Stochastic Gates)**: Uses Gaussian-based continuous relaxation of Bernoulli distributions for stable gradient estimation (Yamada et al., 2020)
2. **STE (State-Through Estimator)**: Binary gates with gradient approximation (Bengio, Léonard and Courville, 2013)
3. **Gumbel Softmax**: Categorical relaxation for feature gating (Herrmann, Bowen and Zabih, 2020)
4. **Correlated STG**: Extension for handling correlated features (Lee, Imrie and Schaar, 2022)

Each method can be used independently or in combination, depending on the specific requirements of the task. Conceptual diagram showing feature gating mechanism.

2. Implemented Methods

This section provides detailed descriptions of the four feature selection methods implemented in SToG, including their theoretical foundations, algorithms, and practical considerations.

2.1 STG: Stochastic Gates with Gaussian Relaxation

The STG method, proposed by Yamada et al. (ICML 2020), addresses the challenge of optimizing the 0 norm—the count of non-zero features—which is combinatorially hard and non-differentiable. STG introduces a probabilistic approach by modeling feature selection as sampling from independent Bernoulli distributions. Each feature x_d has an associated Bernoulli gate with probability μ_d, which determines whether the feature is active. By using a continuous relaxation, STG makes this process fully differentiable, enabling gradient-based optimization.

Let $\mathbf{x} \in \mathbb{R}^D$ be the input features and y be the target variable. The goal is to learn a function $f_\theta(\mathbf{x} \odot \mathbf{z})$ that predicts y while using only a small subset of features, where $\mathbf{z} \in \{0, 1\}^D$ is a binary vector indicating which features are selected.

For each feature dimension $d = 1, \dots, D$, the stochastic gate is defined as:

$$z_d = \max(0, \min(1, \mu_d + \epsilon_d))$$

where μ_d is a learnable parameter controlling the gate probability, $\epsilon_d \sim \mathcal{N}(0, \sigma^2)$ is Gaussian noise with fixed variance σ^2, the max-min operation creates a "hard sigmoid" that clips values to [0,1].

This formulation approximates sampling from a Bernoulli distribution with probability $p_d = \Phi(\mu_d/\sigma)$, where Φ is the standard Gaussian CDF.

The training objective combines a task-specific loss with a regularization term encouraging sparsity:

$$\mathcal{L}(\theta, \mu) = \mathbb{E}_{\mathbf{z} \sim p_\mu} [L(f_\theta(\mathbf{x} \odot \mathbf{z}), y)] + \lambda \sum_{d=1}^{D} \Phi\left(\frac{\mu_d}{\sigma}\right)$$

where $L(\cdot)$ is the task loss (cross-entropy for classification, MSE for regression), λ is the regularization strength hyperparameter, $\Phi(\mu_d/\sigma)$ approximates the expected value of z_d, representing the probability that feature d is active.

2.2 Straight-Through Estimator

The **straight-through (ST) estimator** is a simple heuristic that treats the hard threshold as if it were an identity function during backpropagation:

$$\text{Forward pass: } h = \mathbb{1}_{a>0}$$

$$\text{Backward pass: } \frac{\partial L}{\partial a} = \frac{\partial L}{\partial h}$$

In other words, gradients are copied directly through the threshold without modification. This is clearly biased (the true gradient is zero), but it is simple and often works well in practice.

2.3 Gumbel Softmax for Channel Selection

While STG and SEFS focus on feature selection at the input layer, the Gumbel Softmax method by Herrmann et al. addresses channel selection within neural network layers. This approach has two main applications:

1. Network Pruning: permanently remove unnecessary channels to create a smaller, faster model
2. Conditional Computation: dynamically decide which channels to compute based on the input, enabling adaptive inference

Both applications aim to reduce computational cost while maintaining accuracy. Nevertheless, it is also possible to use Gumbel Softmax relaxation in the task of features selection at the input layer.

To sample from a categorical distribution with probabilities $\mathbf{p} = (p_1, \ldots, p_K)$, Gumbel-Max trick can be used:

$$z = \arg\max_k \left(\log p_k + G_k \right)$$

where $G_k \sim \text{Gumbel}(0, 1)$ are i.i.d. Gumbel random variables.

We get Gumbel Softmax relaxation by replacing the $\arg\max$ with a softmax,:

$$z_k = \frac{\exp\left((\log p_k + G_k)/\tau \right)}{\sum_{j=1}^{K} \exp\left((\log p_j + G_j)/\tau \right)}$$

where $\tau > 0$ is a temperature parameter. As $\tau \to 0$, the softmax approaches a one-hot vector. For binary gates (keep/remove a channel):

$$z = \sigma\left(\frac{1}{\tau} \left(\log \frac{p}{1-p} + G \right) \right)$$

where p is the probability of keeping the channel, $G = G_1 - G_2$ with $G_1, G_2 \sim \text{Gumbel}(0, 1)$, and $\sigma(\cdot)$ is the sigmoid function.

2.4 Correlated stochastic gating

The method introduces correlated gating via Gaussian copula to handle multicollinear features—a common issue in high-dimensional datasets where features are highly correlated. Independent gating mechanisms can select redundant features, while correlated gating encourages competition among related features.

The method models the gate vector $\mathbf{m}$ as a multivariate Bernoulli distribution whose correlation structure is derived from the input features:

1. *Correlation Matrix*: Compute $\mathbf{R}$ from the training data:

$$R_{ij} = \text{Corr}(X_i, X_j)$$

2. *Gaussian Copula*: Generate correlated uniform variables $\mathbf{u} = (u_1, \ldots, u_D)$ via:

$$\mathbf{u} = \Phi_{\mathbf{R}}(\mathbf{v})$$

where $\mathbf{v} \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$ and $\Phi_{\mathbf{R}}$ is the Gaussian copula CDF.

3. *Multivariate Bernoulli Sampling*: Sample gates as:

$$z_d = \mathbb{1}_{u_d < \pi_d}$$

Reparameterization: For differentiability, use a continuous relaxation:

$$z_d = \sigma\left(\frac{1}{\tau} \left(\log \pi_d - \log(1 - \pi_d) + \log u_d - \log(1 - u_d) \right) \right)$$

where $\sigma(\cdot)$ is the sigmoid function and τ is a temperature parameter.

The optimized objective is:

$$\mathcal{L}(\theta, \pi) = \mathbb{E}_{\mathbf{z} \sim p_\pi} \left[L(f_\theta(\mathbf{x} \odot \mathbf{z}), y) \right] + \lambda \sum_{d=1}^{p} \pi_d.$$

3. Library Architecture

We were analysing two approaches to realisation of the library: *a)* solver based: realising feature selection method as separate solvers *b)* layer based: realising feature selection methods as separate layers. Finally, we have chosen the second approach, because it has several advantages:

1. a user has an opportunity to choose layers where he wants to add stochastic gates,

2. theoretically, it provides an option of combining different kinds of feature selection layers in one neural network.

The first point allows us to use the library both for feature selection task (using gates only at the beggining of neural network) and channel reduction task (using gates after every parametrized layer).

We demonstrate the usage and performance of realised methods on feature selection task and channels reduction tasks.

4. Demonstration

To demonstrate the usage of our library on feature selection task, we run all the methods on breast cancer dataset. We add feature selection layer to the beginning of three layer neural network with ReLU activations and train the whole network with different λ-s, accounting for the importance of regularization loss in optimization problem. To reduce the dispersion of the results we make several (10) runs of each method and each λ. We present the results of experiment on Fig.1:
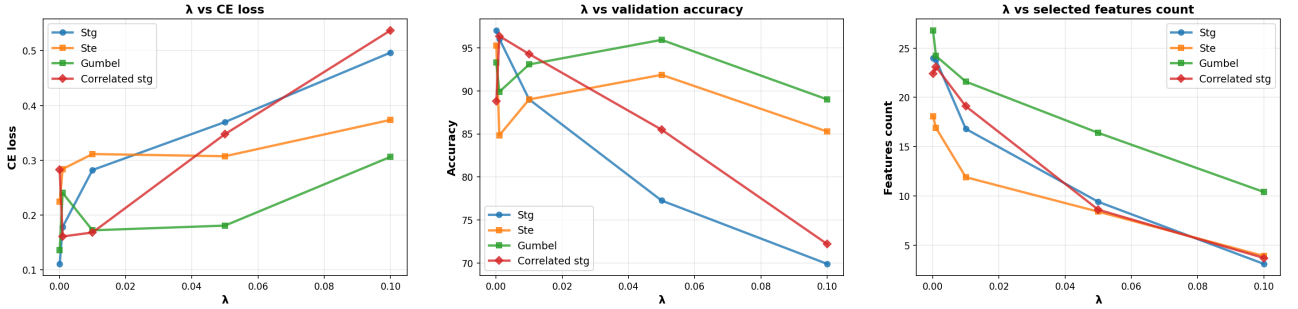

Figure 1: *a)* Cross-Entropy loss vs λ for realised methods, *b)* validation accuracy vs λ for realised methods, *c)* Number of selected features vs λ for realised methods.

For channel reduction task, we also run all the methods on breast cancer dataset. Here, we add feature selection layers at the beginning neural network and after each activation, except the last. We present the results of experiment on Fig.2:
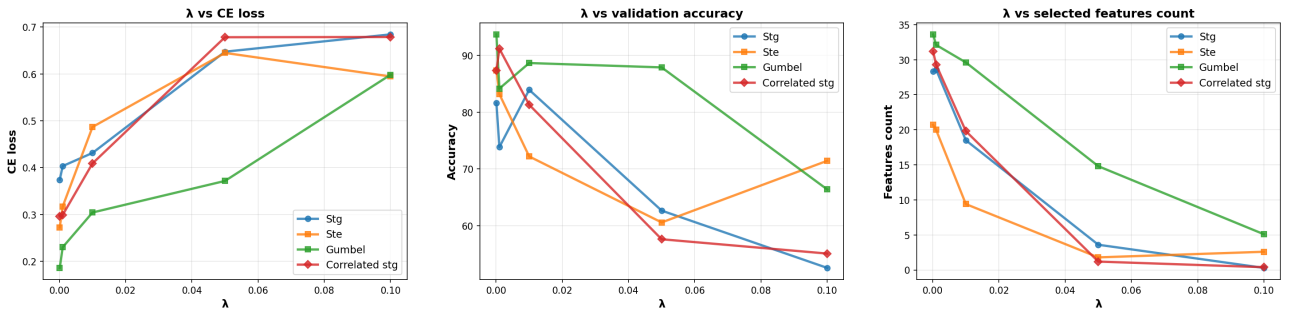

Figure 2: Channels reduction task *a)* cross-entropy loss vs λ for realised methods, *b)* validation accuracy vs λ for realised methods, *c)* Number of selected features vs λ for realised methods.

As we can see from both graphs, using GumbelLayer leaves more features, compared to other methods. There are no conclusions that can be drawn from this observation, since the examples are only demonstrative.

5. Conclusion

We implemented a python library SToG for end-to-end stochastic feature selection in neural networks. The library enables end-to-end differentiable feature selection through stochastic gating mechanisms, allowing neural

networks to simultaneously learn predictive models while identifying the most relevant features. Different stochastic gating approaches implemented as separate layers that are compatible with torch. Because of this realisation our library can be easily used both for feature selection and channels reduction tasks. We demonstrate the usage of the SToG library and share details that worth noting when using library in practise.

References

Bengio, Y., Léonard, N. and Courville, A., 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432.*

Herrmann, C., Bowen, R.S. and Zabih, R., 2020. Channel selection using gumbel softmax. In: *European conference on computer vision.* Springer.pp.241–257.

Lee, C., Imrie, F. and Schaar, M. van der, 2022. Self-supervision enhanced feature selection with correlated gates. In: *International conference on learning representations.*

Yamada, Y., Lindenbaum, O., Negahban, S. and Kluger, Y., 2020. Feature selection using stochastic gates. In: *International conference on machine learning.* PMLR.pp.10648–10659.