Нейронные сети для табличных данных

Классификация нейронных сетей для табличных данных

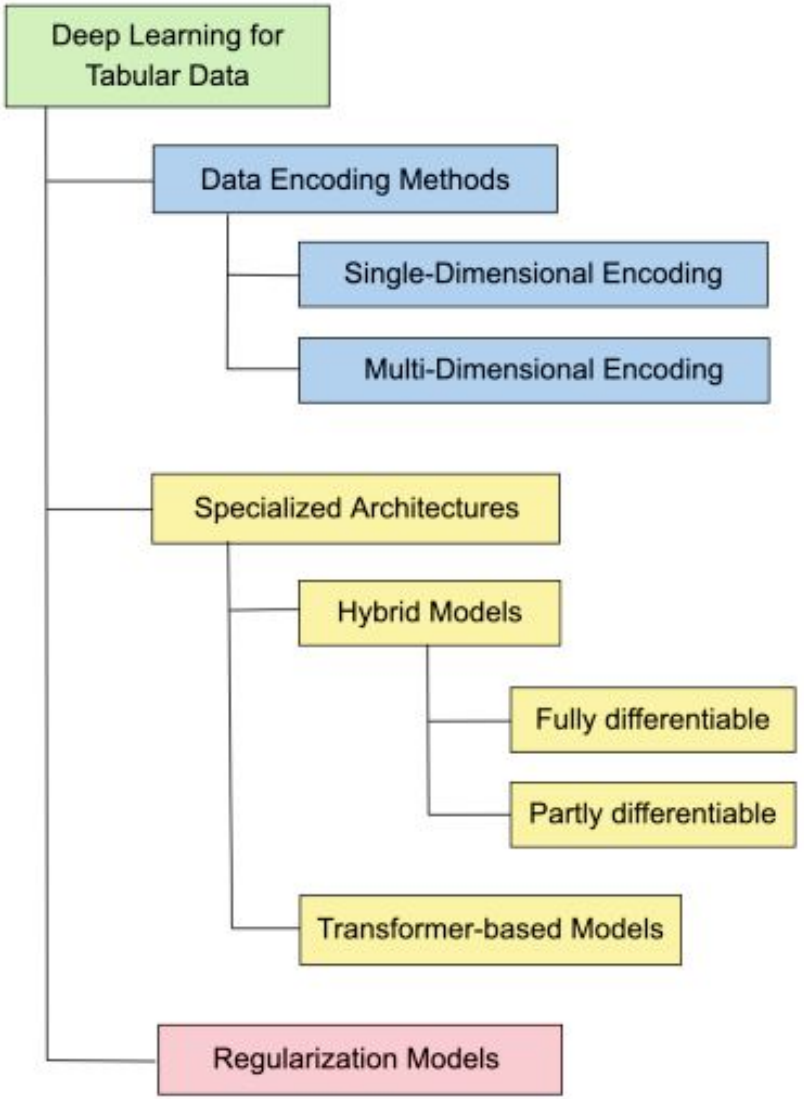


Figure 1: Unified taxonomy of deep neural network models for heterogeneous tabular data.

Vadim Borisova et al., 2021. Deep Neural Networks and Tabular Data: A Survey.

	Method	Interpr.	Key Characteristics
Encoding	SuperTML Sun et al. (2019)		Transform tabular data into images for CNNs
	VIME Yoon et al. (2020)	✓	Self-supervised learning and contextual embedding
	IGTD Zhu et al. (2021)		Transform tabular data into images for CNNs
	SCARF Bahri et al. (2021)		Self-supervised contrastive learning
Architectures, Hybrid	Wide&Deep Cheng et al. (2016)		Embedding layer for categorical features
	DeepFM Guo et al. (2017)		Factorization machine for categorical data
	SDT Frosst and Hinton (2017)	✓	Distill neural network into interpretable decision tree
	xDeepFM Lian et al. (2018)		Compressed interaction network
	TabNN Ke et al. (2018)		DNNs based on feature groups distilled from GBDT
	DeepGBM Ke et al. (2019)		Two DNNs, distill knowlegde from decision tree
	NODE Popov et al. (2019)		Differentiable oblivious decision trees ensemble
	NON Luo et al. (2020)		Network-on-network model
	DNN2LR Liu et al. (2020)		Calculate cross feature wields with DNNs for LR
	Net-DNF Katzir et al. (2021)		Structure based on disjunctive normal form
	Boost-GNN Ivanov and Prokhorenkova (2021)		GNN on top decision trees from the GBDT algorithm
	SDTR Luo et al. (2021)		Hierarchical differentiable neural regression model

Architectures, Transformer	TabNet Arik and Pfister (2019)	✓	Sequential attention structure
	TabTransformer Huang et al. (2020)	✓	Transformer network for categorical data
	SAINT Somepalli et al. (2021)	✓	Attention over both rows and columns
	ARM-Net Cai et al. (2021)		Adaptive relational modeling with multi-headgated attention network
Regular.	RLN Shavitt and Segal (2018)	✓	Hyperparameters regularization scheme
	Regularized DNNs Kadra et al. (2021)		A "cocktail" of regularization techniques

Agenda

- Factorization Machine
- Deep FM
- TabNet

Factorization Machine

$$\hat{y}(x) := w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle \boldsymbol{v}_i, \boldsymbol{v}_j \rangle x_i x_j$$

$$w_0 \in R,\, w \in R^n,\, \boldsymbol{v} \in R^{n \times k}$$

- w_0 - смещение (bias);
- w_i - определяет влияние каждой характеристики по отдельности;
- $\langle \boldsymbol{v}_i, \boldsymbol{v}_j \rangle$ - попарное взаимодействие двух характеристик (второго порядка). То есть вместо того, чтобы сделать один вес для каждой пары, модель делает именно факторизацию обучаемых параметров каждой характеристики и по отдельности строит связи.

https://www.csie.ntu.edu.tw/~b97053/paper/Rendle2010FM.pdf

Risk
modelling.research

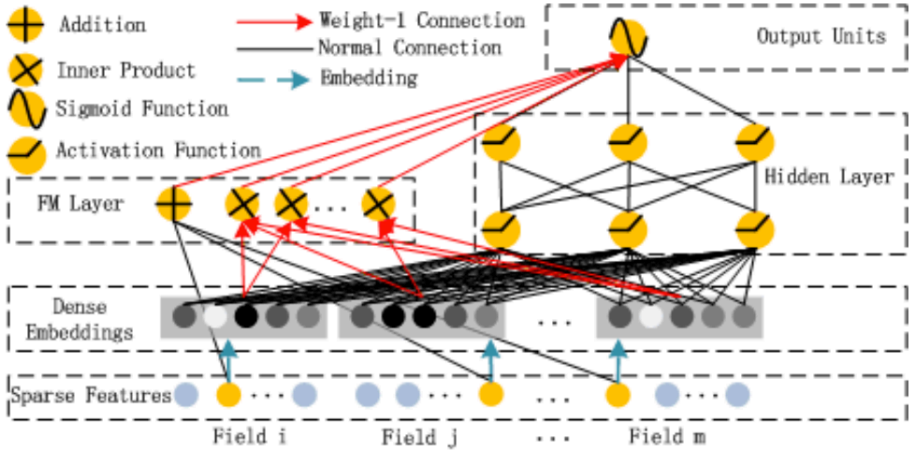
Factorization Machine

Сложность вычислений для второго порядка $O(kn)$

$$
\begin{aligned}
&\sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle \mathbf{v}_i, \mathbf{v}_j \rangle \, x_i \, x_j \\
=&\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \langle \mathbf{v}_i, \mathbf{v}_j \rangle \, x_i \, x_j - \frac{1}{2} \sum_{i=1}^{n} \langle \mathbf{v}_i, \mathbf{v}_i \rangle \, x_i \, x_i \\
=&\frac{1}{2} \left(\sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{f=1}^{k} v_{i,f} \, v_{j,f} \, x_i \, x_j - \sum_{i=1}^{n} \sum_{f=1}^{k} v_{i,f} \, v_{i,f} \, x_i \, x_i \right) \\
=&\frac{1}{2} \sum_{f=1}^{k} \left(\left(\sum_{i=1}^{n} v_{i,f} \, x_i \right) \left(\sum_{j=1}^{n} v_{j,f} \, x_j \right) - \sum_{i=1}^{n} v_{i,f}^2 \, x_i^2 \right) \\
=&\frac{1}{2} \sum_{f=1}^{k} \left(\left(\sum_{i=1}^{n} v_{i,f} \, x_i \right)^2 - \sum_{i=1}^{n} v_{i,f}^2 \, x_i^2 \right)
\end{aligned}
$$

Risk
modelling.research

Deep Factorization Machine (DeepFM)



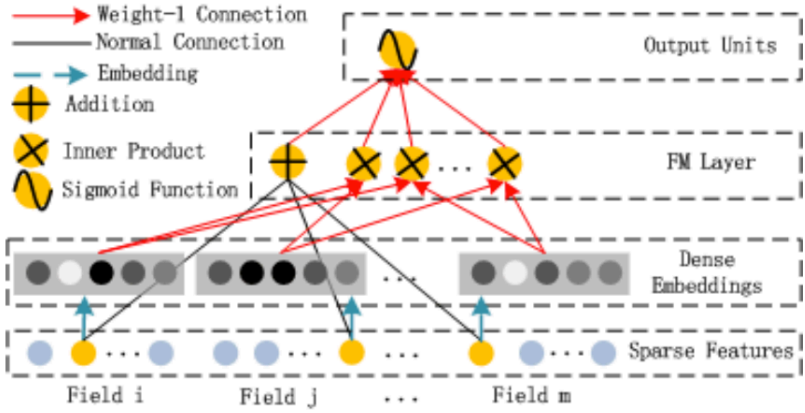
https://arxiv.org/pdf/1703.04247.pdf

Deep Factorization Machine (DeepFM)

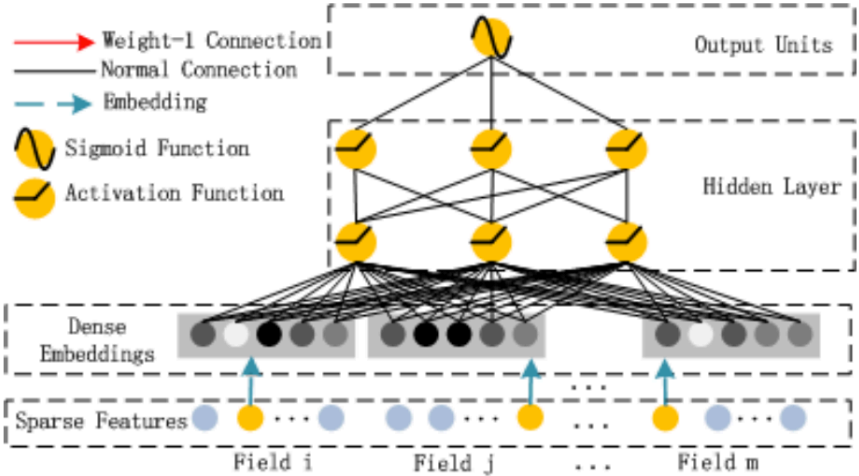
Модель учит объяснять таргет на основе "low- and high-order feature interactions".

$$\hat{y} = sigmoid(y_{FM} + y_{DNN})$$

FM component



Deep component



Deep component

Embedding Layer:

$$a^{(0)} = [e_1, e_2, ..., e_m],$$

Hidden Layer:

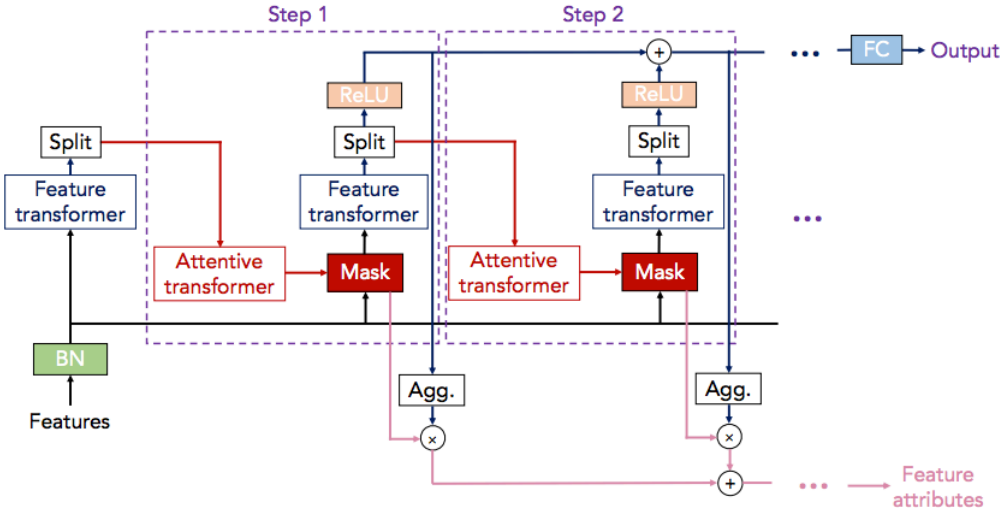
$$a^{(l+1)} = sigmoid(W^{(l)}a^{(l)} + b^{(l)})$$

Выход:

$$y_{DNN} = sigmoid(W^{|H|+1}a^{H} + b^{|H|+1}),$$

где $|H|$ - кол-во скрытых слоев.

TabNet



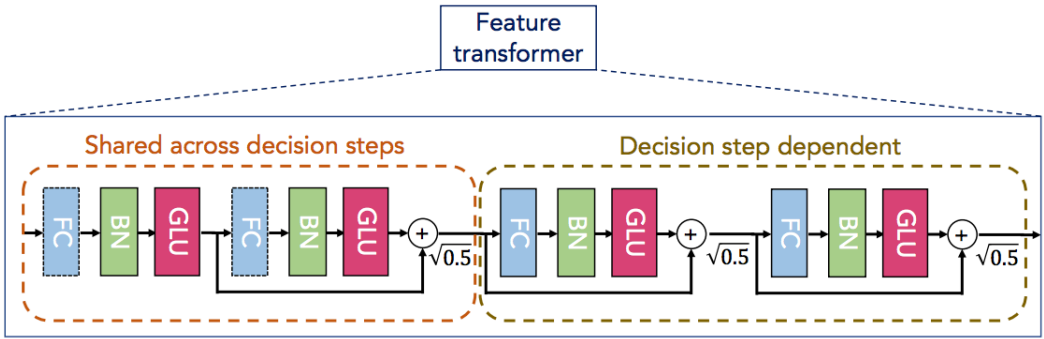
https://arxiv.org/pdf/1908.07442.pdf

Risk modelling.research

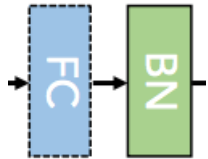
TabNet

Предлагает нейронную сеть с последовательным механизмом внимания для интерпретации результата.

Объединяем полносвязные слои в блоки



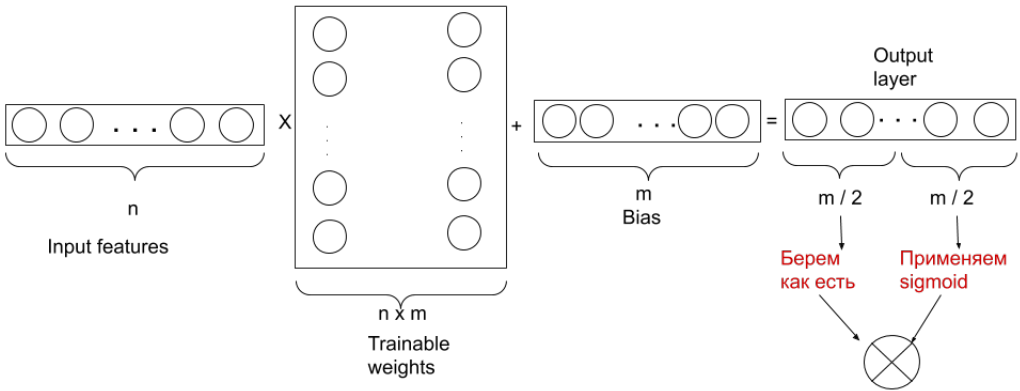
Linear + Batch Norm



torch.nn.Linear

torch.nn.BatchNorm1d

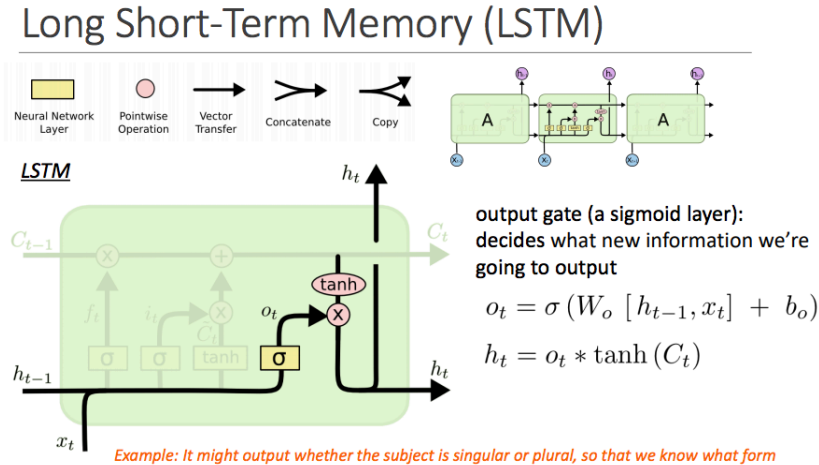
GLU -- Gated Linear Unit



https://arxiv.org/pdf/1612.08083.pdf

Gating Mechanism

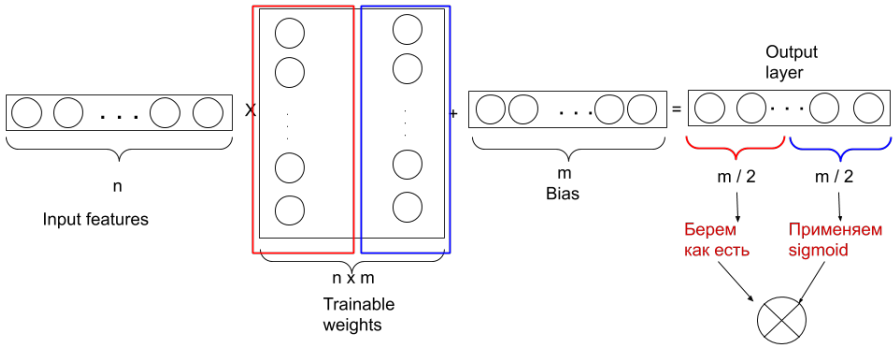
Механизм, свойственный для рекуррентных нейронных сетей.



Gating Mechanism

$$h_t = tanh(X \cdot W + b) \otimes \sigma(X \cdot V + c)$$

Gating Mechanism



Gating Mechanism

Gated Tanh Unit

$$\nabla[tanh(X) \otimes \sigma(X)] = tanh'(X)\nabla X \otimes \sigma(X) + \sigma'(X)\nabla X \otimes tanh(X)$$

Уменьшающие коэффициенты в виде $tanh'(X), \sigma'(X)$, которые могут привести к эффекту исчезающих градиентов по мере добавления слоев.

Gating Mechanism

Gated Linear Unit

$$\nabla[X \otimes \sigma(X)] = \nabla X \otimes \sigma(X) + X \otimes \sigma'(X)\nabla X$$

В таком подходе перед $\sigma(X)$ нет уменьшающего коэффициента, и это считается как мультипликативный skip connection, который наименее склонен к эффекту исчезающих градиентов по мере добавления слоев. Приводит к быстрой сходимости.

Split

Выходной тензор из **Feature Transformer** делим на две части:

$$[d[i], a[i]] = f_i(M[i] \cdot f)$$

$$d[i] \in R^{B \times N_d}$$

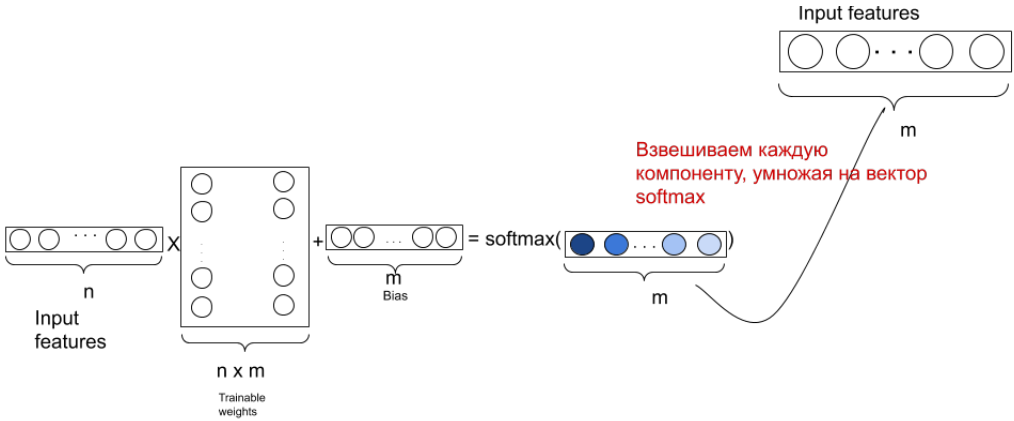
и

$$a[i] \in R^{B \times N_a}$$

Первая часть $d[i]$ идет на агрегацию с другими выходами от каждого шага.
Вторая часть $a[i]$ идет для вычисления маски -- в **Attentive Transformer**.

Risk
modelling.research

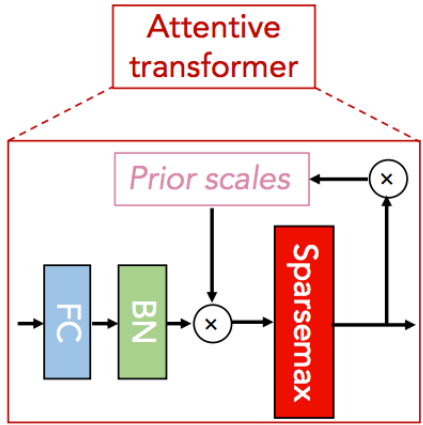
Механизм внимания



Механизм внимания

Влияние механизма внимания происходит при backpropagation -- обновлении весов.
Так как происходит произведение на тензор, следовательно, каждый соответсвующий вес, который участвовал в вычислении тензора, получает свой дополнительный множитель.

Attentive Transformer - Механизм внимания в TabNet



Sparsemax аналог Softmax

$Sparsemax$ исправляет недостаток $Softmax$: $Softmax$ никогда не примет значение ноль.
И для тех случаев, где нужно получить разреженное распределение вероятности,$Softmax$ не подходит.

Sparsemax

Идея заключается в том, чтобы найти такое пороговое значение, которое бы позволило обнулить некоторые значения и оставить отличные от нуля другие.

Sparsemax

$$\text{sparsemax}_i(\boldsymbol{z}) = [z_i - \tau(\boldsymbol{z})]_+$$

$\tau(z)$ - пороговая функция, которая определяет, что обнулить, а что оставить.

Sparsemax

Algorithm 1 Sparsemax Evaluation

Input: $\boldsymbol{z}$

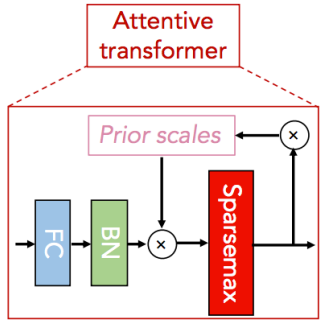
Sort $\boldsymbol{z}$ as $z_{(1)} \geq \ldots \geq z_{(K)}$

Find $k(\boldsymbol{z}) := \max \left\{ k \in [K] \mid 1 + k z_{(k)} > \sum_{j \leq k} z_{(j)} \right\}$

Define $\tau(\boldsymbol{z}) = \frac{\left(\sum_{j \leq k(\boldsymbol{z})} z_{(j)}\right) - 1}{k(\boldsymbol{z})}$

Output: $\boldsymbol{p}$ s.t. $p_i = [z_i - \tau(\boldsymbol{z})]_+$.

Механизм внимания в TabNet. Prior Scale



$$M[i] = sparsemax(P[i-1] \cdot h_i(a[i-1])),$$

где i - номер шага, $h_i(a[i-1])$ выход после BatchNorm, $P[i-1]$ - Prior Scale

Prior Scale - как часто использовалась характеристика до текущего шага

$$P[i] = \prod_{j=1}^{i} (\gamma - M[j])$$

, где γ - параметр релаксации,
с увеличением γ характеристике будет придаваться бОльший вес.

Маска

$$M_{b,j}[i] = 0$$

Это значит j-я характеристика в батче b для i-го наблюдения не оказалась значительной.

Важно то, что маска вычисляется для каждого шага и наблюдения отдельная. В статье не предложен способ получения агрегированной маски по всем наблюдениям, но можно получить агрегированную для всех шагов (decision steps).

Маска

$$M_{agg-b,j} = \sum_{i=1}^{N_{steps}} \eta_b[i] M_{b,j}[i] / \sum_{j=1}^{D} \sum_{i=1}^{N_{steps}} \eta_b[i] M_{b,j}[i]$$

$$\eta_b[i] = \sum_{c=1}^{N_d} ReLU(d_{b,c}[i])$$

Если $d_{b,c}[i] < 0$, то значит все характеристики на i-м шаге не будут влиять на агрегированное решение.

Маска

Целевая функция для контролирования степени разреженности маски

$$L_{sparse} = \sum_{i=1}^{N_{steps}} \sum_{b=1}^{B} \sum_{j=1}^{D} \frac{-M_{b,j}[i] log(M_{b,j}[i] + \varepsilon)}{N_{steps} \cdot B}$$

Суммируется с основной целевой функцией с коэффициентом λ_{sparse}