Model structure

MIPT

2022

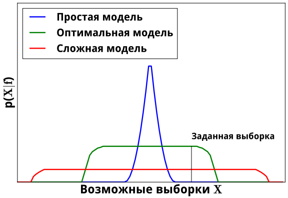
Model selection

First level: select optimal parameters:

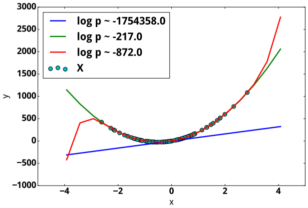
$$\mathrm{w} = \arg\max \frac{p(\mathfrak{D}|\mathrm{w})p(\mathrm{w}|\mathrm{h})}{p(\mathfrak{D}|\mathrm{h})},$$

Second level: select model optimizing Evidence:

$$p(\mathfrak{D}|\mathrm{h}) = \int_{\mathrm{w}} p(\mathfrak{D}|\mathrm{w})p(\mathrm{w}|\mathrm{h})d\mathrm{w}.$$

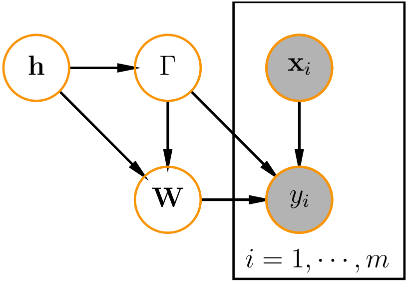


Model selection scheme



Example

Prior



Definition

Prior for parameters w and structure $\mathbf{\Gamma}$ of the model f is a distrubution $p(\mathrm{W}, \mathbf{\Gamma}|\mathrm{h}) : \mathbb{W} \times \Gamma \times \mathbb{H} \to \mathbb{R}^+$, where $\mathbb{W}$ is a parameter space, Γ is a structure space.

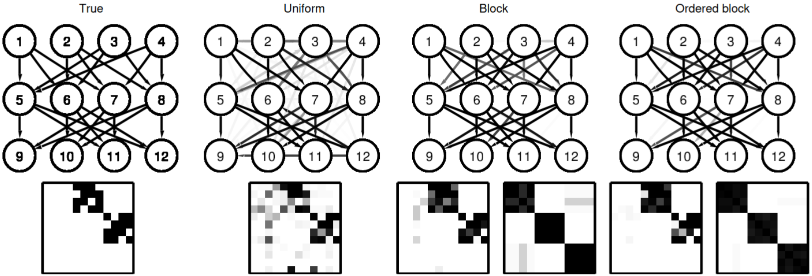
Definition

Hyperparameters $\mathrm{h} \in \mathbb{H}$ of the models are the parameters of $p(\mathrm{w}, \mathbf{\Gamma}|\mathrm{h})$ (parameters of prior f).

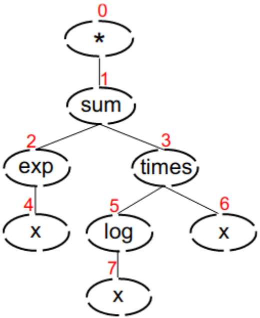
Model f is assigned by the following values:

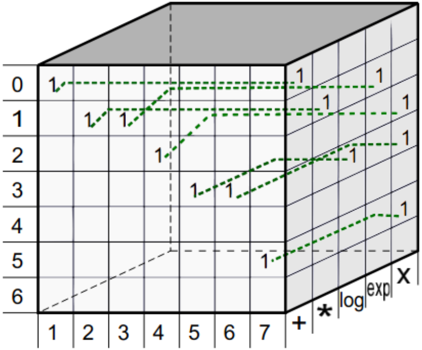
- **Parameters** $\mathrm{w} \in \mathbb{W}$ set the superposition of submodels f_v.
- **Structure parameters** $\mathbf{\Gamma} = \{\gamma^{j,k}\}_{(j,k) \in E} \in \Gamma$ set importance of each submodel f_v.
- **Hyperparameters** $\mathrm{h} \in \mathbb{H}$ set prior distribution .
- **Metaparameters** $\boldsymbol{\lambda} \in \Lambda$ set optimization function.

Example: Bayesian networks



Example: prediction of ranking functions





$f = \exp(x) + (\log x)x$

Трехиндексная матрица связей Z_f дерева Γ_f

- вершины дерева пронумерованы;
- первые два индекса — номера вершин в ребре;
- третий индекс — выбранная элементарная функция на конце ребра.

Optimal Brain Damage

The problem of removing unrelevant parameters (pruning) is considered.
Idea: Consider Taylor series for maximum point $\boldsymbol{\theta}^*$:

$$L(\boldsymbol{\theta}^* + \Delta\boldsymbol{\theta}) - L(\boldsymbol{\theta}^*) = -\frac{1}{2}\boldsymbol{\theta}^{\mathsf{T}}\mathsf{H}\boldsymbol{\theta} + o(||\Delta\boldsymbol{\theta}||^3),$$

where H is Hessian of $-L$.
Diagonalize the Hessian:

$$L(\boldsymbol{\theta}^* + \Delta\boldsymbol{\theta}) \to \max$$

where

$$\theta_i^* + \Delta\theta_i = 0.$$

Relevance of parameter:

$$\frac{\theta_i^2}{2[\mathsf{H}^{-1}]_{i,i}}.$$

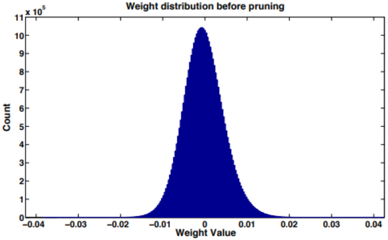
Learning both Weights and Connections for Efficient Neural Networks

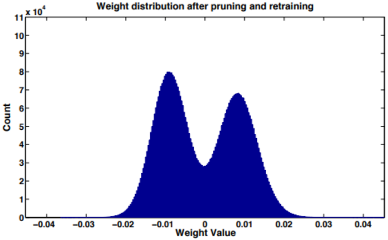
Idea:

1. Optimize model;
2. Remove parameters with minimal magnitude;
3. Repeat optimization.

Near-obvious facts that can be found in the article:

- L_2 is better for pruning than L_1 if we repeat optimization.
- It's better to re-optimize from the previous optimum than from random start.
- The parameter distribution becomse multimodal after pruning.



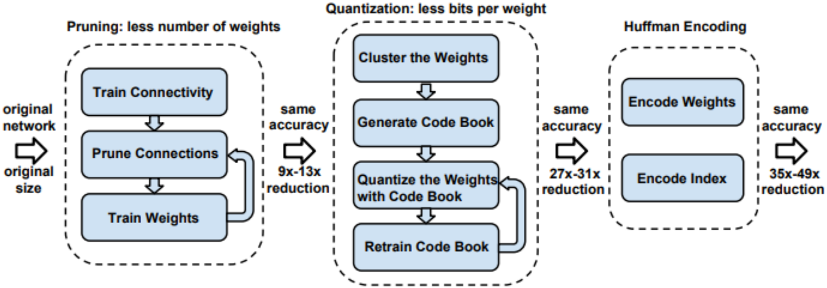


Deep Compression

Idea:

1. Remove unrelevant parameters similar to previous approach.
2. Clusterize parameters (K-means for each layer).
3. Repeat optimization using centroids.
4. Encode parameter indices using Huffman coding scheme.

Result: reduce model size 40x, speedup x3.



Graves, 2011

$$\mathrm{MDL}(\mathrm{f}, \mathfrak{D}) = L(\mathrm{f}) + L(\mathfrak{D}|\mathrm{f}),$$

where f is a model, $\mathfrak{D}$ is a dataset, L is a description length in bits.

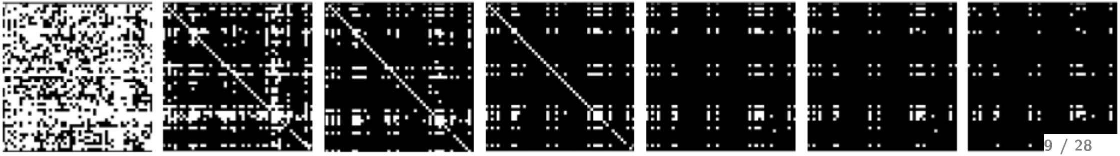
$$\mathrm{MDL}(\mathrm{f}, \mathfrak{D}) \sim L(\mathrm{f}) + L(\mathrm{w}^*|\mathrm{f}) + L(\mathfrak{D}|\mathrm{w}^*, \mathrm{f}),$$

w^* are optimal parameters.

$$L = \sum_{\mathrm{x},\mathrm{y}} \log p(\mathrm{y}|\mathrm{x}, \hat{\mathrm{w}}) + \frac{1}{2}\left(\mathrm{tr}(\mathrm{A}_q) + \boldsymbol{\mu}_q^{\mathrm{T}} \mathrm{A}^{-1} \boldsymbol{\mu}_q - \ln |\mathrm{A}_q|\right).$$

Prune parameters w_i using relative PDF:

$$\lambda = \frac{q(0)}{q(\boldsymbol{\mu}_{i,q})} = \exp\left(-\frac{\mu_i^2}{2\sigma_i^2}\right).$$



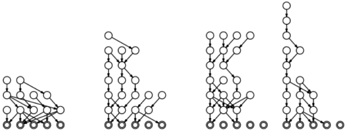
Model generation: example

Adams et al., 2010:

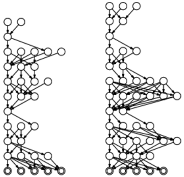
- The problem is to generate Deep belief networks
- The structure **Γ** is a sequence of adjacent matrices for each layer
- Generate structure using Monte-Carlo with Indian Buffet prior with parameters α, β
- Hyperparameter interpretation: width and sparsity of each layer

(a) $\alpha = 1,\ \beta = 1$

(b) $\alpha = \frac{1}{2},\ \beta = 1$



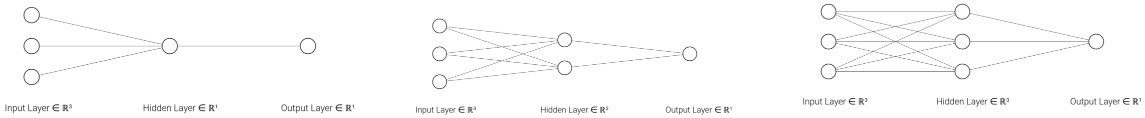
(c) $\alpha = 1,\ \beta = 2$



(d) $\alpha = \frac{3}{2},\ \beta = 1$

Structure selection example

hidden layer dim = 1 | hidden layer dim = 2 | hidden layer dim = 3



All these models can be represented as $f(x, w) = \boldsymbol{\sigma}\left((w^2)^\mathsf{T} \boldsymbol{\sigma}\left((w^1)^\mathsf{T} x\right)\right)$
with similar shape of $w^1 : \dim(w^1) = 3 \times 3$.

Structure selection: one-layer network

The model f is defined by the **structure** $\mathbf{\Gamma} = [\gamma^{0,1}, \gamma^{1,2}]$.

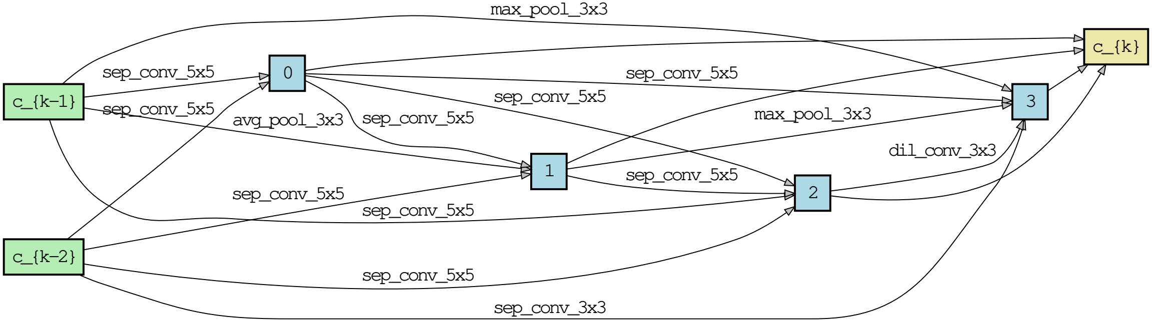
$$\text{Model: } \mathsf{f}(\mathsf{x}) = \textbf{softmax}\left((\mathsf{w}_0^{1,2})^{\mathsf{T}} \mathsf{f}_1(\mathsf{x})\right), \quad \mathsf{f}(\mathsf{x}) : \mathbb{R}^n \to [0,1]^{|\mathbb{Y}|}, \quad \mathsf{x} \in \mathbb{R}^n.$$

$$\mathsf{f}_1(\mathsf{x}) = \gamma_0^{0,1} \mathsf{g}_0^{0,1}(\mathsf{x}) + \gamma_1^{0,1} \mathsf{g}_1^{0,1}(\mathsf{x}),$$

where $\mathsf{w} = [\mathsf{w}_0^{0,1}, \mathsf{w}_1^{0,1}, \mathsf{w}_0^{1,2}]^{\mathsf{T}}$ — parameter matrices, $\{\mathsf{g}^0_{0,1}, \mathsf{g}^1_{0,1}, \mathsf{g}^0_{1,2}\}$ — generalized-linear functions, alternatives of layers of the network.

$$\begin{tikzcd}
\mathsf{f}_0(\mathsf{x}) = \mathsf{x} \arrow[r, bend left, "\gamma_0^{0,1} \mathsf{g}_0^{0,1}(\mathsf{x}) = \gamma_0^{0,1} \boldsymbol{\sigma}\left((\mathsf{w}_0^{0,1})^{\mathsf{T}} \mathsf{x}\right)"] \arrow[r, bend right, "\gamma_1^{0,1} \mathsf{g}_1^{0,1}(\mathsf{x}) = \gamma_1^{0,1} \boldsymbol{\sigma}\left((\mathsf{w}_1^{0,1})^{\mathsf{T}} \mathsf{x}\right)"'] & \mathsf{f}_1(\mathsf{x}) \arrow[r, "\gamma_0^{1,2} \mathsf{g}_0^{1,2}(\mathsf{x}) = \gamma_0^{1,2} \textbf{softmax}\left((\mathsf{w}_0^{1,2})^{\mathsf{T}} \mathsf{x}\right)"] & \mathsf{f}_2(\mathsf{x})
\end{tikzcd}$$

Neural architecture search example



Structure selection: neural architecture search space

The model f is defined by the **structure** $\boldsymbol{\Gamma} = [\gamma^{0,1}, \gamma^{1,2}]$.

$$\text{Model: } \mathrm{f}(\mathrm{x}) = \textbf{softmax}\left((\mathrm{w}_0^{1,2})^{\mathsf{T}} \mathrm{f}_1(\mathrm{x})\right), \quad \mathrm{f}(\mathrm{x}) : \mathbb{R}^n \to [0,1]^{|\mathbb{Y}|}, \quad \mathrm{x} \in \mathbb{R}^n.$$

$$\mathrm{f}_1(\mathrm{x}) = \gamma_0^{0,1} \mathrm{g}_0^{0,1}(\mathrm{x}) + \gamma_1^{0,1} \mathrm{g}_1^{0,1}(\mathrm{x}),$$

where $\mathrm{w} = [\mathrm{w}_0^{0,1}, \mathrm{w}_0^{1,2}]^{\mathsf{T}}$ — parameter matrices, $\mathrm{g}_{0,1}^0$ is a convolution,$\mathrm{g}_{0,1}^1$ is a pooling operation, $\mathrm{g}_{1,2}^0$ is a generalized-linear function.

$$\begin{tikzcd}
\mathrm{f}_0(\mathrm{x}) = \mathrm{x} \arrow[rr, bend left, "\gamma_0^{0,1} \mathrm{g}_0^{0,1}(\mathrm{x}) = \gamma_0^{0,1} \textbf{Conv}(\mathrm{x}, \mathrm{w}_0^{0,1})"] \arrow[rr, bend right, "\gamma_1^{0,1} \mathrm{g}_1^{0,1}(\mathrm{x}) = \gamma_1^{0,1} \textbf{MaxPool}(\mathrm{x})"'] & & \mathrm{f}_1(\mathrm{x}) \arrow[r, "\gamma_0^{1,2} \mathrm{g}_0^{1,2}(\mathrm{x}) = \gamma_0^{1,2} \textbf{softmax}\left((\mathrm{w}_0^{1,2})^{\mathsf{T}} \mathrm{x}\right)"] & \mathrm{f}_2(\mathrm{x})
\end{tikzcd}$$

Deep learning model structure as a graph

Define:

1. acyclic graph (V, E);
2. for each edge $(j, k) \in E$: a vector primitive differentiable functions $\mathrm{g}^{j,k} = [\mathrm{g}_0^{j,k}, \dots, \mathrm{g}_{K^{j,k}}^{j,k}]$ with length of $K^{j,k}$;
3. for each vertex $v \in V$: a differentiable aggregation function $\mathbf{agg}_v$.
4. a function $\mathrm{f} = \mathrm{f}_{|V|-1}$:

$$\mathrm{f}_v(\mathrm{w}, \mathrm{x}) = \mathbf{agg}_v \left(\{ \langle \boldsymbol{\gamma}^{j,k}, \mathrm{g}^{j,k} \rangle \circ \mathrm{f}_j(\mathrm{x}) | j \in \mathrm{Adj}(v_k) \} \right), v \in \{1, \dots, |V| - 1\}, \quad \mathrm{f}_0(\mathrm{x}) = \mathrm{x} \tag{1}$$

that is a function from $\mathbb{X}$ into a set of labels $\mathbb{Y}$ for any value of $\boldsymbol{\gamma}^{j,k} \in [0, 1]^{K^{j,k}}$.

Definition

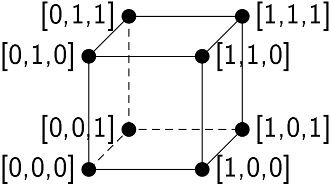
A *parametric set of models* $\mathfrak{F}$ is a graph (V, E) with a set of primitive functions $\{\mathrm{g}^{j,k}, (j, k) \in E\}$ and aggregation functions $\{\mathbf{agg}_v, v \in V\}$.

Statement

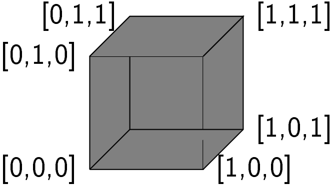
A function $\mathrm{f} \in \mathfrak{F}$ is a model for each $\boldsymbol{\gamma}^{j,k} \in [0, 1]^{K^{j,k}}$.

Structure restrictions

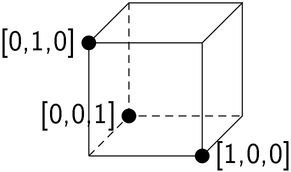
An example of restrictions for structure parameter γ, $|\gamma| = 3$.



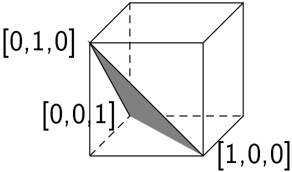
Cube vertices



Cube interior



Simplex vertices

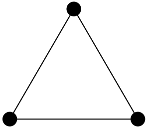


Simplex interior

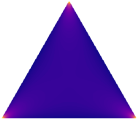
Prior distribution for the model structure

Every point in a simplex defines a model.

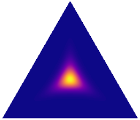
Gumbel-Softmax distribution: $\boldsymbol{\Gamma} \sim \mathrm{GS}(\mathbf{s}, \lambda_{\text{temp}})$



$\lambda_{\text{temp}} \to 0$

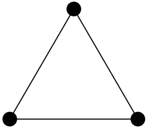


$\lambda_{\text{temp}} = 0.995$

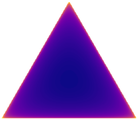


$\lambda_{\text{temp}} = 5.0$

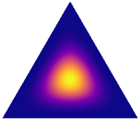
Dirichlet distribution: $\boldsymbol{\Gamma} \sim \mathrm{Dir}(\mathbf{s}, \lambda_{\text{temp}})$



$\lambda_{\text{temp}} \to 0$



$\lambda_{\text{temp}} = 0.995$



$\lambda_{\text{temp}} = 5.0$

Invertible Gaussian reparametrization

$$p(\mathrm{w}) = \bar{\mathrm{softmax}}(\boldsymbol{\alpha}), \quad \boldsymbol{\alpha} \sim \mathcal{N},$$

(there should be a ϵ in the denominator for invertibility of the function)

- Reparameterization works well
- $KL(\mathrm{w}_1|\mathrm{w}_2) = KL(\boldsymbol{\alpha}_1|\boldsymbol{\alpha}_2)$
- Poor interpretation

Neural Architecture Search: problem statement

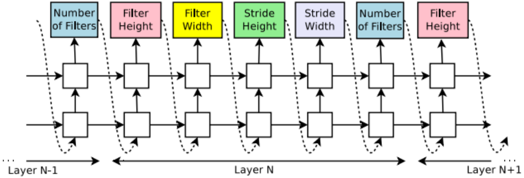
w are model parameters.
$\mathbf{\Gamma}$ is a structure.

$$\mathbf{\Gamma}^* = \arg\max Q(\mathrm{w}^*, \mathbf{\Gamma}),$$

$$\mathrm{w}^* = \arg\max L(\mathrm{w}, \mathbf{\Gamma}).$$

Neural Architecture Search with Reinforcement Learning

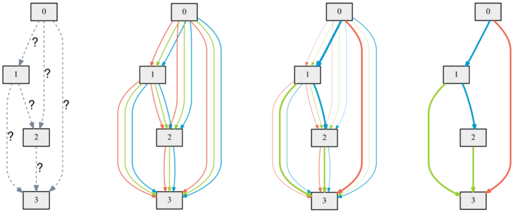
The structure is selected using controllet.
The optimization of model parameters is conducted in a loop of structure selection.



DARTS

The model is a multigraph, where edges $[g^e]$ correspond to submodels, vertices $f_v(x)$ are the results of submodels:

$$f_v = \langle \gamma, \mathbf{softmax}([g^e(x)]) \rangle.$$



DARTS

Optimization:

$$\boldsymbol{\Gamma}^* = \arg\max Q(\mathrm{w}^*, \boldsymbol{\Gamma}),$$

$$\mathrm{w}^* = \arg\max L(\mathrm{w}, \boldsymbol{\Gamma}).$$

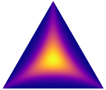
The optimization is done using greedy gradient-like optimization:

$$\nabla_{\boldsymbol{\Gamma}} Q(\mathrm{w}', \boldsymbol{\Gamma}) = \lambda_L \nabla_{\boldsymbol{\Gamma},\mathrm{w}} L(\mathrm{w}, \boldsymbol{\Gamma}) \nabla_{\mathrm{w}} Q(\boldsymbol{\Gamma}, \mathrm{w}').$$

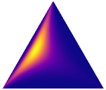
Exhaustive search

Exhaustive search can be done using regularization:

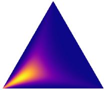
$$\lambda_1 \mathrm{KL}(\mathbf{\Gamma}|\mathbf{\Gamma}_1) + \lambda_2 \mathrm{KL}(\mathbf{\Gamma}|\mathbf{\Gamma}_2) + \ldots$$



$\boldsymbol{\lambda}_{\text{struct}} = [0; 0; 0]$.



$\boldsymbol{\lambda}_{\text{struct}} = [1; 0; 0]$.



$\boldsymbol{\lambda}_{\text{struct}} = [1; 1; 0]$.

Performance criteria for structure selection

- Number of parameters
- Number of vertives
- Number of edges
- Complexity of subfunctions

FBNet

$$\min_{\mathbf{\Gamma}} \min_{\mathbf{w}} L \cdot \lambda_1 \log \mathrm{LAT}(\mathbf{\Gamma}),$$

where LAT is a function of hardware latency of the operations for **target hardware**.

FBNet

Model	#Parameters	#FLOPs	Latency on iPhone X	Latency on Samsung S8	Top-1 acc (%)
FBNet-iPhoneX	4.47M	322M	19.84 ms (target)	23.33 ms	73.20
FBNet-S8	4.43M	293M	27.53 ms	22.12 ms (target)	73.27

Table 5. FBNets searched for different devices.

NAS with complexity control

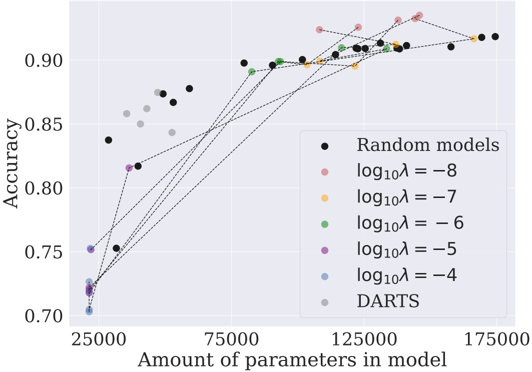
Our proposal

To use a mapping $\boldsymbol{\gamma}(\lambda)$ instead of constant structural parameters $\boldsymbol{\gamma}(\lambda)$, where λ is a regularization term for the loss function:

$$\mathsf{E}_\lambda \left(\log p(\mathsf{y}|\mathsf{X}, \mathsf{w}, \boldsymbol{\Gamma}(\lambda)) + \lambda \sum_{(i,j)} \langle \textbf{softmax}\left(\boldsymbol{\gamma}(\boldsymbol{\lambda})^{(i,j)}\right), \mathsf{n}(\mathsf{g}^{(i,j)}) \rangle \right),$$

where $\mathsf{n}(\mathsf{g}^{(i,j)})$ is a vector of amount of parameters for all the basic functions g.

Example: CIFAR-10



$$\mathsf{E}_\lambda \left(\log p(\mathsf{y}|\mathsf{X}, \mathsf{w}, \mathbf{\Gamma}(\lambda)) + \lambda \sum_{(i,j)} \langle \mathbf{softmax}\left(\boldsymbol{\gamma}(\boldsymbol{\lambda})^{(i,j)}\right), \mathsf{n}(\mathsf{g}^{(i,j)}) \rangle \right).$$

Reference

- Bishop C. M., Nasrabadi N. M. Pattern recognition and machine learning. – New York : springer, 2006. – Т. 4. – №. 4. – С. 738.
- Mansinghka V. et al. Structured priors for structure learning //arXiv preprint arXiv:1206.6852. – 2012.
- Варфоломеева А. А. Методы структурного обучения в задаче обобщения структур прогностических моделей, магистерская диссертация.
- LeCun Y., Denker J., Solla S. Optimal brain damage //Advances in neural information processing systems. – 1989. – Т. 2.
- Han S. et al. Learning both weights and connections for efficient neural network //Advances in neural information processing systems. – 2015. – Т. 28
- Potapczynski A., Loaiza-Ganem G., Cunningham J. P. Invertible gaussian reparameterization: Revisiting the gumbel-softmax //arXiv preprint arXiv:1912.09588. – 2019.
- Graves A. Practical variational inference for neural networks //Advances in neural information processing systems. – 2011. – Т. 24.
- Han S., Mao H., Dally W. J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding //arXiv preprint arXiv:1510.00149. – 2015.
- Adams R. P., Wallach H., Ghahramani Z. Learning the structure of deep sparse graphical models //Proceedings of the thirteenth international conference on artificial intelligence and statistics. – JMLR Workshop and Conference Proceedings, 2010. – С. 1-8.
- Jang E., Gu S., Poole B. Categorical reparameterization with gumbel-softmax //arXiv preprint arXiv:1611.01144. – 2016.
- Zoph B., Le Q. V. Neural architecture search with reinforcement learning //arXiv preprint arXiv:1611.01578. – 2016.
- Бахтеев О. Ю. 2020. Байесовский выбор субоптимальной структуры модели глубокого обучения. Диссертация.
- Liu H., Simonyan K., Yang Y. Darts: Differentiable architecture search //arXiv preprint arXiv:1806.09055. – 2018.
- Yakovlev K. D. et al. Neural Architecture Search with Structure Complexity Control //International Conference on Analysis of Images, Social Networks and Texts. – Springer, Cham, 2022. – С. 207-219.
- Wu B. et al. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search //Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. – 2019. – С. 10734-10742.