Deep Neural Networks

A Nonparametric Bayesian View with Local Competition

Sergios Theodoridis[1,2]

[1]Dept. of Informatics and Telecommunications, National and Kapodistrian University of Athens, Athens, Greece.
and
[2]Chinese University of Hong Kong, Shenzhen, China.

Chinese University of Hong Kong, Shenzhen - SRIBD
May 9, 2019

Joined Work with
Soterios Chatzis and Konstantinos Panousis

Outline of the Talk

- Motivations and Goals: Short and Long Term Ones
- Local Competition in Neuronal Activities: The Local Winner-Take-All (LWTA) Unit
- The Framework of our Discussion: The Nonparametric Bayesian Approach and the Indian Buffet Process (IBP)
- The Model: Fully Connected NNs
- The Model: Convolutional Neural Networks
- A World of Probabilities: All you Need is Reparameterization
- The Cost Function: The World is Small After All
- Was is it Worth the Effort?: Some Results
- The Why's: Time for Speculations
- The first results of this line of research are reported in Proceedings ICML 2019

Deep Networks: A Success Story

- Deep NNs have been established as the state-of-the art in Machine Learning. Often, human or even superhuman performance is reported on diverse artificially constructed data sets.
- The multilayer rationale of building a learner/predictor is here to stay.
- The multilayer structure provides a way of efficient representation of the information that resides in the available data.
- However, this great advantage does not come without drawbacks.

Deep Networks: Some Major Drawbacks

- The structural complexity associated with the multilayer rationale poses major challenges in their interpretability. The "how" and the "why" behind a prediction.
- Deciding the architecture, i.e., the required, for the network, set of nodes/convolutions/synapses of a layer is still a painful task for any designer.
- They suffer from the so called catastrophic forgetting. When our brain learns, say a task A, it can generalize and learn a second one, B, without forgetting A. Existing deep networks tend to forget the previous task, A, when they learn the new one, B.

Deep Networks: Some Major Drawbacks

- Currently, their training needs formidable computational power. Scaling deep learning algorithms to low profile hardware remains a challenging task.
- Performing billions of predictions per day comes with substantial energy costs. Graphical Processing Units (GPUs) are energy consuming devices.
- Real-time predictions are estimated, in many cases, to be two orders of magnitude away in terms of speed from what deep NNs can deliver.
- "Communicating" NNs with millions of parameters through band limited channels is still impractical.

Artificial and Brain Neurons

- Conceptually, the basic model that deep networks are built upon does not seem to comply with what a neural network tries to "imitate"; that is, how the brain functions in order to reach decisions.
- Deep networks are based on a convenient mathematical approximation model. On the contrary, the brain neurons are activated via spiking sequences, there exist lateral "connections" among them and they compete for their firing.

Our Short Term Goals

Our short term goals are:

- To develop a **systematic** framework whereby the architecture adapts to the data: As reported before, this includes the number of nodes/convolutions/synapses needed in a layer.
- To develop a methodology that reduces computational requirements and scales better to lower-power devices, such as mobile phones, robots and cars.
- These goals are in line with the well established, by now, fact that a number of currently developed architectures carry a high degree of parameter redundancy, e.g., [1].
- To work with a model that bears a closer inspirational "affinity" with the way our "brain" operates. Experimental evidence in neuroscience has shown that the brain works in a probabilistic way, as it deals with information involving uncertainties.

Our Longer Term Goals

Our longer term goals are:

- Attack the problem of catastrophic forgetting. Extend and develop our technique to cope with continuous learning tasks, e.g., in the context of communications, robotics, reinforcement learning.
- Exploit the specific structure of our proposed model for hardware implementations, since, in principle, it lends itself to spiking networks and/or the use of memristors ([2], [3]).

The Two Innovative Aspects of the Proposed Approach

- The work brings together two concepts:
  1. Replace the nonlinearity used in deep networks (squashing functions, ReLUs) with the biologically inspired Local Winner-Take-All mechanism (LWTA).
  2. Adopt a fully Bayesian framework for the parameter estimation. Specifically, the nonparametric Bayesian framework will be adopted and the Indian Buffet Process (IBP) will be employed to impose priors on the theoretically infinite number of parameters/nodes.

The Local Winner-Take-All Unit

- **Competitive** interactions among neurons and neural circuits have long played an important role in biological models of brain processes.
- Experimental evidence indicates that many cortical and sub-cortical regions of the brain exhibit a recurrent on-center, off-surround anatomy. That is, cells provide **excitatory** feedback to nearby cells, while scattering **inhibitory** signals over a broader range, e.g. [4], [5], [6].
- This takes place in the brain in either of two forms: a) analogue values are encoded through firing rates and the most frequently firing neuron exerts the strongest inhibition on its competitors, or b) analogue values are encoded to timing delays, where the earliest firing neuron (corresponding to the largest value) inhibits its competitors from firing.

The Local Winner-Take-All Unit

- In simple words: Due to competition, i.e. based on local/lateral information, some of the neurons "raise" in the excitatory state and their outputs are passed over to other neurons. The rest are inhibited and remain "silent".
- The simplest way to model competition is via an LWTA unit. This comprises U linear neurons, with a common input, $\boldsymbol{x}$. Each linear neuron computes a weighted average over the input excitations, i.e. it outputs

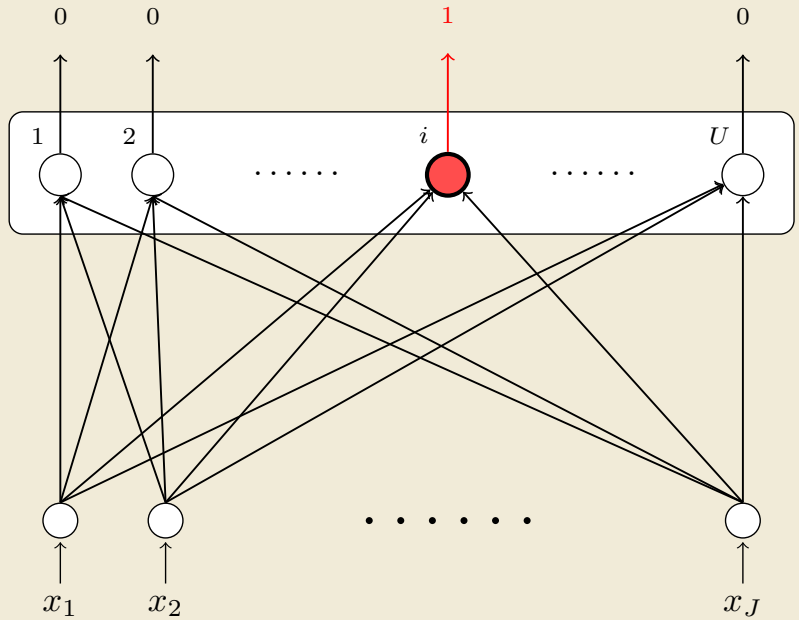
$$h_u = \boldsymbol{w}_u^T \boldsymbol{x}, \; u = 1, 2, \ldots, U.$$

Then, the corresponding U outputs of the LTWA are

$$\xi_u = \begin{cases} 1, & \text{if } h_u > h_j, \; j = 1, 2, \ldots, U, j \neq u, \\ 0, & \text{otherwise.} \end{cases}$$

- That is, only the neuron with the strongest response to the common input to the LWTA fires an 1 and and the rest are inhibited to "silence", i.e., to the 0 value.

The Local Winner-Take-All Unit



The neuron in red has the strongest response and it is the one that fires.

The Local Winner-Take-All Unit

- It has been shown in [7] that a single-layer neural network where the nodes are (a soft version of the previous) LTWA units is a universal approximator.
- Besides the above basic version, other versions have also been proposed. In the sequel, we adopt the following model.

$$y_u = \xi_u h_u, \ \xi_u \in \{0, 1\}$$

- In our work, the values for the ξ_u's are obtained via fully probabilistic arguments.

The Bayesian Framework

- In the Bayesian framework, the unknown parameters are treated as random variables. Thus, they are described in terms of:
  - A prior distribution: It encapsulates our uncertainty, before the observations have been taken into account. This is provided by the user. It corresponds to a regularizer.
  - A posterior distribution: It quantifies our uncertainty about the random parameters after the observations have been obtained.
  - Both, the prior and the posterior are selected to be of a known specific functional parameterised form. Their unknown (hyper)parameters are optimized during training.
- In simple words, in Bayesian learning, the task of training comprises the estimation of the unknown hyper(parameters).
- Typically, a variant of the EM algorithm is mobilized.

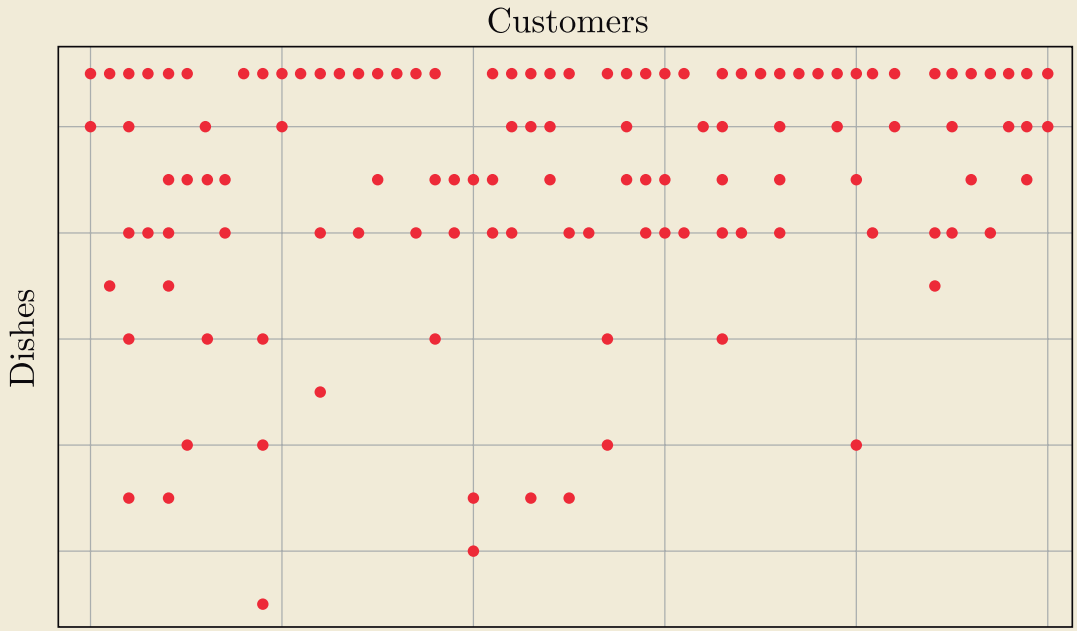
The Bayesian Framework

- In the parametric Bayesian framework, the number of model parameters is a-priori fixed. Hence, standard priors are utilized, e.g., Gaussian, Discrete, etc.
- In the nonparametric Bayesian context, the number of some of the random parameters/variables is not fixed. The number is **learned** from the data. In theory, the number of the corresponding parameters is assumed to be a-priori **infinite**. The estimated number of parameters is obtained via their posterior distributions.
- Thus, in such cases, priors for infinite number of parameters are required.
- The Chinese Restaurant Process and the Indian Buffet Process are typical priors associated with infinitely many parameters. The former has been used in clustering and the latter for Matrix Factorization.

The Indian Buffet Process (IBP)

- IBP has been developed as a prior on infinite binary random matrices. The prior, in line with the rationale that runs sparsity-inducing priors, promotes sparsity, by generating matrices that have many zeros.
- The name (IBP) draws from the parallel of Indian restaurants that offer many (infinite) dishes. Cuctomers/data choose dishes/features. If a dish is selected, this action corresponds to 1, and if it is not selected, it corresponds to 0. This is how the infinite matrix is filled with ones and zeros.
- The crucial point that promotes **sparsity** is that the probability of choosing any dish, which has already been selected before by another customer, may become lower than the probability of the new customer to choose a new dish.

The Indian Buffet Process (IBP)



- One can think of the dots as ones and their absence as zeros. Observe that the matrix is mainly filled by zeros.
- In the ML framework, customers are observations and dishes correspond to characteristics/features. The figure indicates which of the observations share certain features.

The Indian Buffet Process: The Stick Breaking Construction

- For each customer/observation, n, the allocation of 1's takes place according to the following rule:
  - Choose a beta distribution, $\text{Beta}(\beta|\alpha, 1)$.
  - Sample $\beta_1 \sim \text{Beta}(\beta|\alpha, 1)$ and set $\pi_1 = \beta_1$.
  - Then, the kth step of the algorithm is given by,

$$\begin{aligned} \beta_k &\sim \text{Beta}(\beta|\alpha, 1), \\ \pi_k &= \prod_{j=1}^{k} \beta_j, \ k = 1, 2, \ldots \end{aligned}$$

- Assign the corresponding binary matrix element, $z_{kn} = 1$ according to the draw,

$$z_{kn} \sim \text{Bern}(z|\pi_k), \ k = 1, 2, \ldots$$

The Indian Buffet Process: The Stick Breaking Construction

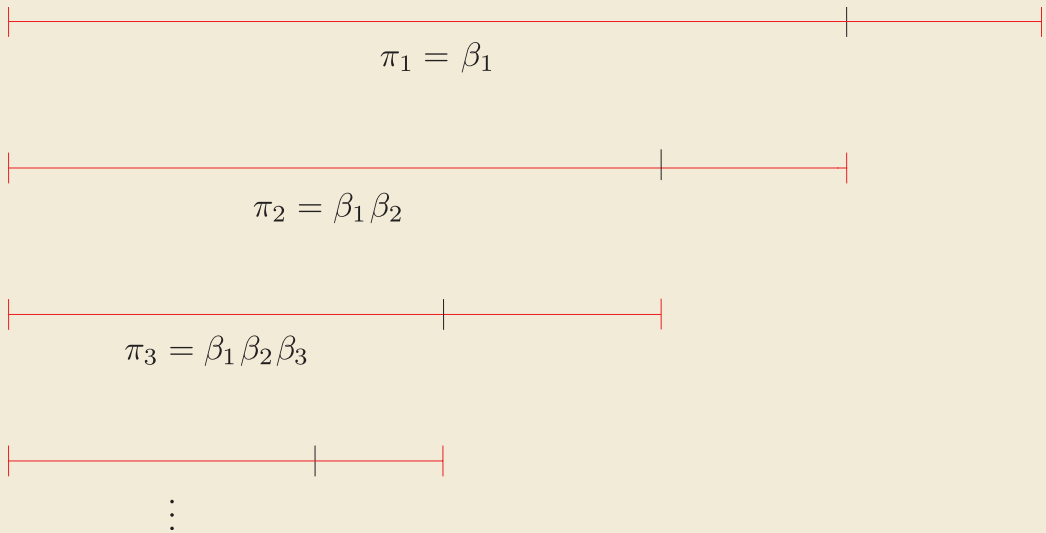
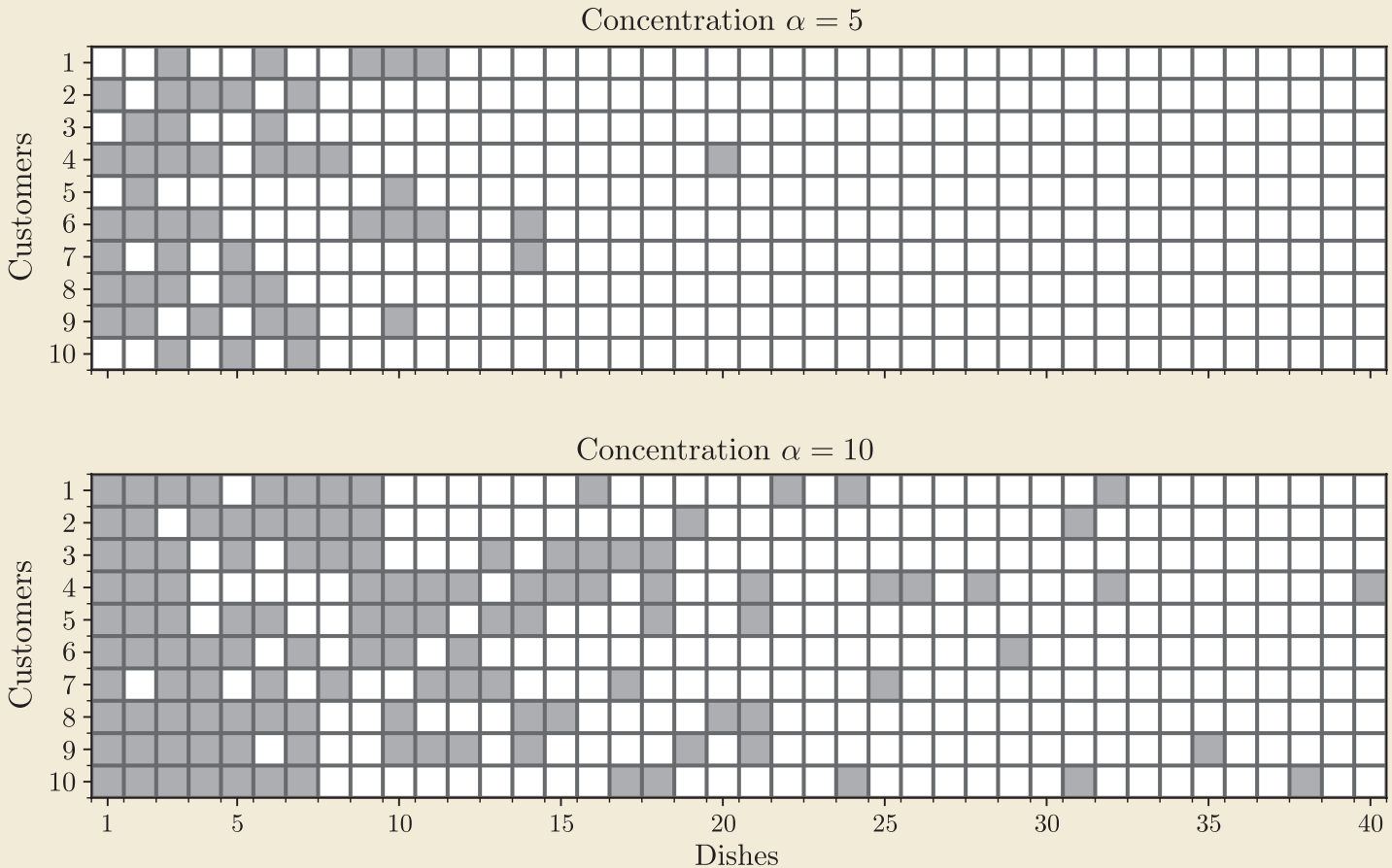


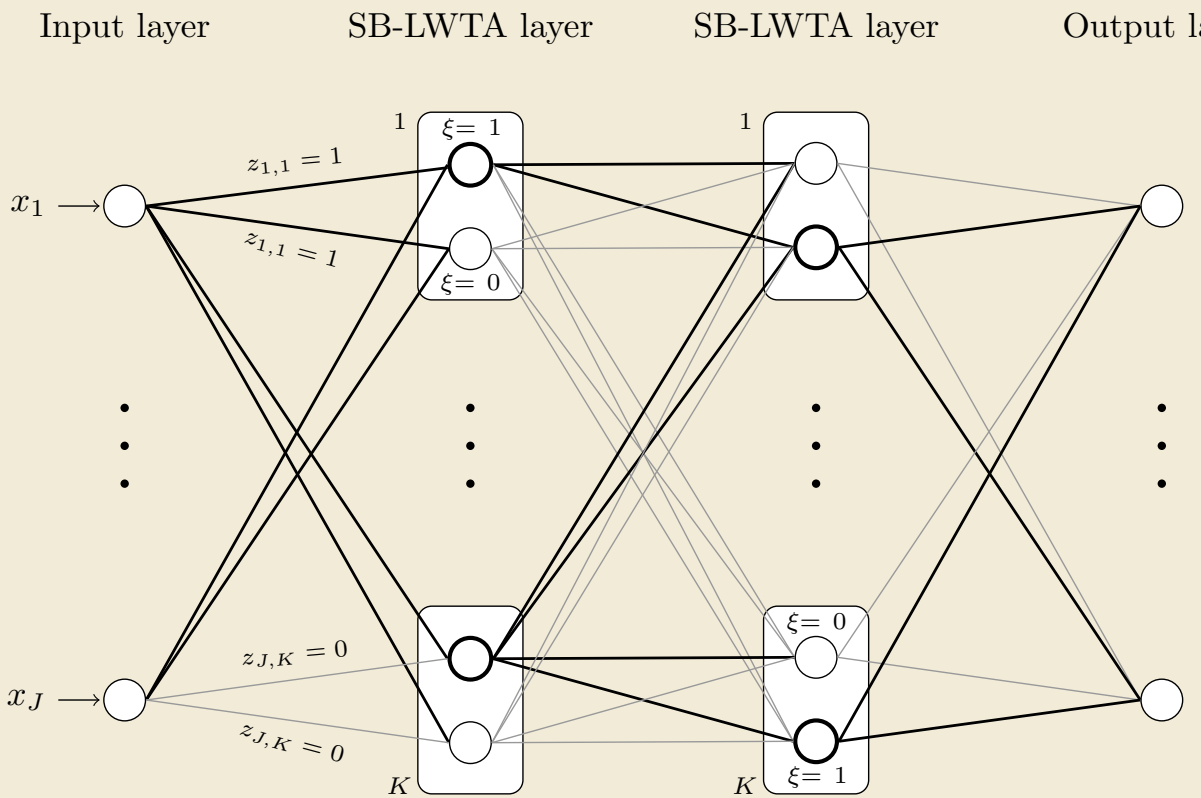
Illustration of the IBP stick-breaking construction.

The Stick Breaking Construction: The Dependence on the Parameter α



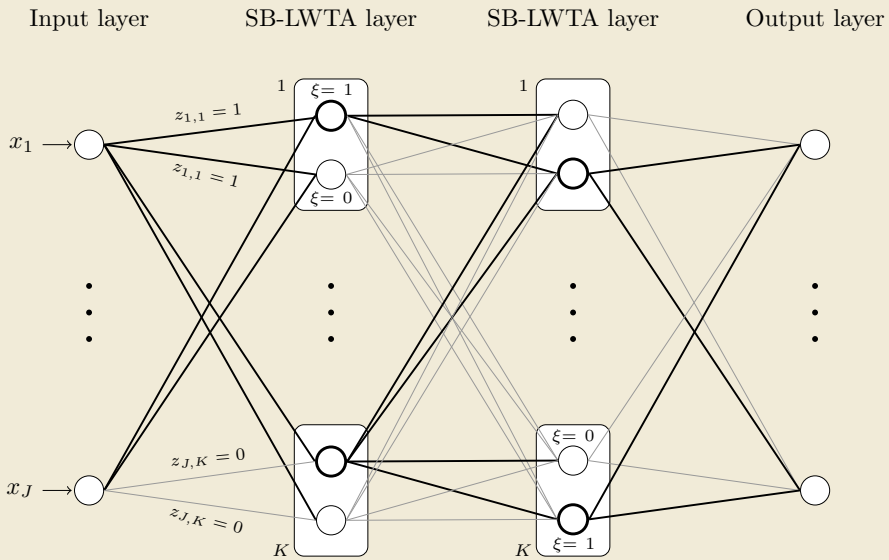
Dependence of the IBP on the parameter α in Beta$(\beta|\alpha, 1)$. The smaller α is the sparser the prior matrix becomes.

The Nonparametric Bayesian Feedforward Fully Connected NN



The Nonparametric Bayesian Feedforward Fully Connected NN

- The involved random parameters that describe the network:



- The weights of all the synapses:
  $W:\ w_{j,k,u},\ j|_1^J, k|_1^K, u|_1^U,$
- The hidden variables $z_{j,k},\ j|_1^J,\ k|_1^K,$
  $Z:\ z_{j,k} \in \{0,1\}$, i.e., binary r.v.
- The latent variables $\xi_{k,u},\ k|_1^K,\ u|_1^U.$
  $\boldsymbol{\xi}_k \in \mathbb{R}^U$ one-hot vectors.

- Each observation pair, $(\boldsymbol{y}_n^o, \boldsymbol{x}_n) \in \mathbb{R}^M \times \mathbb{R}^J$, where M is the number of output classes, is associated with a corresponding set of latent variables, i.e. $\boldsymbol{\xi}_{n,k}$, $n = 1, 2, \ldots, N$, for each layer.

Prior and Posterior Distributions of the RVs

- The weights:
  - The priors: Gaussians

$$W \sim \prod_{j,k,u} \mathcal{N}\left(w_{j,k,u}|0,1\right).$$

  - The posteriors: Gaussians

$$q(W) = \prod_{j,k,u} \mathcal{N}\left(w_{j,k,u}|\mu_{j,k,u}, \sigma^2_{j,k,u}\right)$$

Prior and Posterior Distributions of the RVs

- The Z hidden utility variables:
  - The prior: IBP

    For the kth LWTA do:

    $$\beta_k \sim \text{Beta}(\alpha, 1),\ \pi_k = \prod_{i=1}^{k} \beta_i,\ z_{j,k} \sim \text{Bern}(z|\pi_k),\ j = 1, 2, \ldots, J$$

  - The posteriors: Bernoulli

    $$q(z_{j,k}) \sim \text{Bern}\left(z|\tilde{\pi}_{j,k}\right)$$

Prior and Posterior Distributions of the RVs

- The latent indicator variables $\boldsymbol{\xi}$:
  - The priors: Categorical

$$\boldsymbol{\xi}_{n,k} \sim \text{Cat}\left(\boldsymbol{\xi}\middle|\frac{1}{U},\ldots,\frac{1}{U}\right)$$

  - The posteriors: Categorical

$$q(\boldsymbol{\xi}_{n,k}) = \text{Cat}\left(\boldsymbol{\xi}\middle|P_{n,k,1},\ldots,P_{n,k,U}\right),\ k=1,2,\ldots$$

    where the softmax nonlinearity has been employed, i.e.

$$P_{n,k,u} = \frac{\exp(h_{n,k,u})}{\sum_{u=1}^{U}\exp(h_{n,k,u})},$$

    and the responses of the U neurons in the kth LWTA are the linear combinations of the inputs,

$$h_{n,k,u} = \sum_{j=1}^{J}(w_{j,k,u}\cdot z_{j,k})x_{n,j}.$$

Prior and Posterior Distributions of the RVs

- In words, the lateral interaction among the neurons in every LWTA is implicitly imposed via the use of the softmax nonlinearity.
- The neuron that wins and fires is **probabilistically** decided according to a probability distribution that relates all U neurons. This is computed based on the values of their individual responses. This step comprises the lateral connection among the linear neurons within the LWTA.
- Translating values to probabilities is in line with our Bayesian framework.

Reparameterization of the Random Variables

- In the standard form of the neural networks, optimization takes place with respect to all the parameters. Starting from some initial conditions, an iterative scheme is used for their updates.
- In the Bayesian setting, the parameters are used for the forward pass, yet the optimization takes place with respect to the hyperparameters that define the respective random variables, e.g., mean values, variances, probabilities.
- However, drawing values randomly from the respective distributions does not involve the hyperparameters of the distributions in an explicit form to be used in the optimization. To this end, the so called reparameterization trick is employed.
- The random variables are transformed and expressed in terms of the hyperparameters that define the respective distributions.

The Random Variables Reparameterized

- The Gaussian weights, are reparameterized via their mean and variances,

$$w = \mu + \sigma\epsilon,\ \epsilon \sim \mathcal{N}(0, 1).$$

- The hidden variables, i.e., $z_{j,k}$, are reparameterized according to the Gumbel-Softmax relaxation trick for discrete variables (e.g. [8]). If $\tilde{\pi}_{j,k}$ is the corresponding probability of $z_{j,k}$, $k = 1, 2, \ldots,$, then

$$z_{j,k} = \frac{1}{1 + \exp(-(\ln \tilde{\pi}_{j,k}) + g_{j,k})/\lambda)} \tag{1}$$

$$g_{j,k} = \ln \epsilon_{j,k} - \ln(1 - \epsilon_{j,k}),\ \epsilon_{j,k} \sim \text{Uniform}(0, 1),$$

  where the values of λ are annealed during training.

- The same reparameterization method is used for the indicator variables, $\xi_{n,k,u}$.

The Random Variables Reparameterized

- Besides the reparameterization of the weights w's and the utility variables z's, reparameterization is also used for the β_k random variables. These are used for generating samples to estimate expectations during the optimization stage.
- For the posterior of each of the the β_k, the Beta$(\cdot|a_k, b_k)$ distribution is adopted. For its reparameterization, it is first approximated by the Kumaraswamy distribution ([9]),

$$p(\beta_k; a_k, b_k) = a_k b_k \beta_k^{a_k - 1}(1 - \beta_k^{a_k})^{b_k - 1},$$

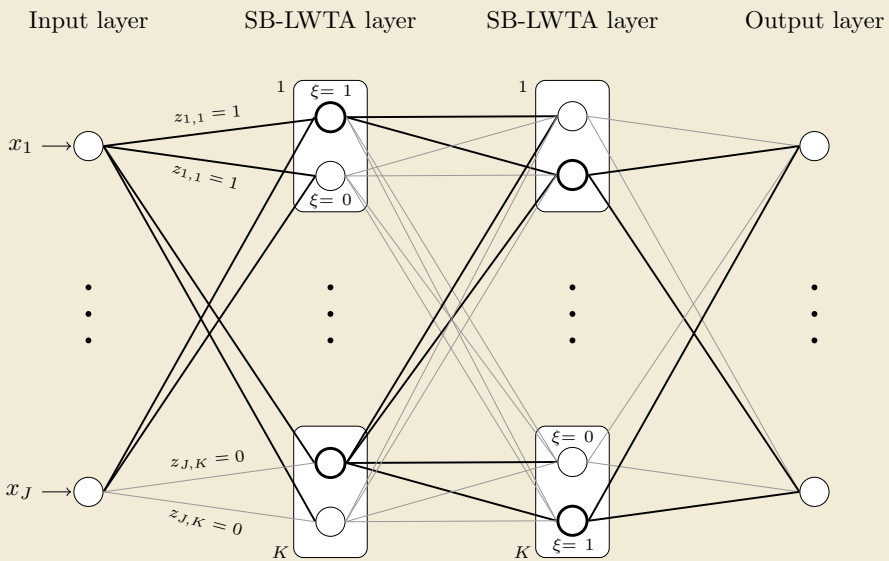
and then samples are drawn from the respective reparameterized form ([10]),

$$\beta_k = \left(1 - (1 - X)^{\frac{1}{b_k}}\right)^{\frac{1}{a_k}}, \; X \sim U(0, 1).$$

The Nonparametric Bayesian Feedforward Fully Connected NN

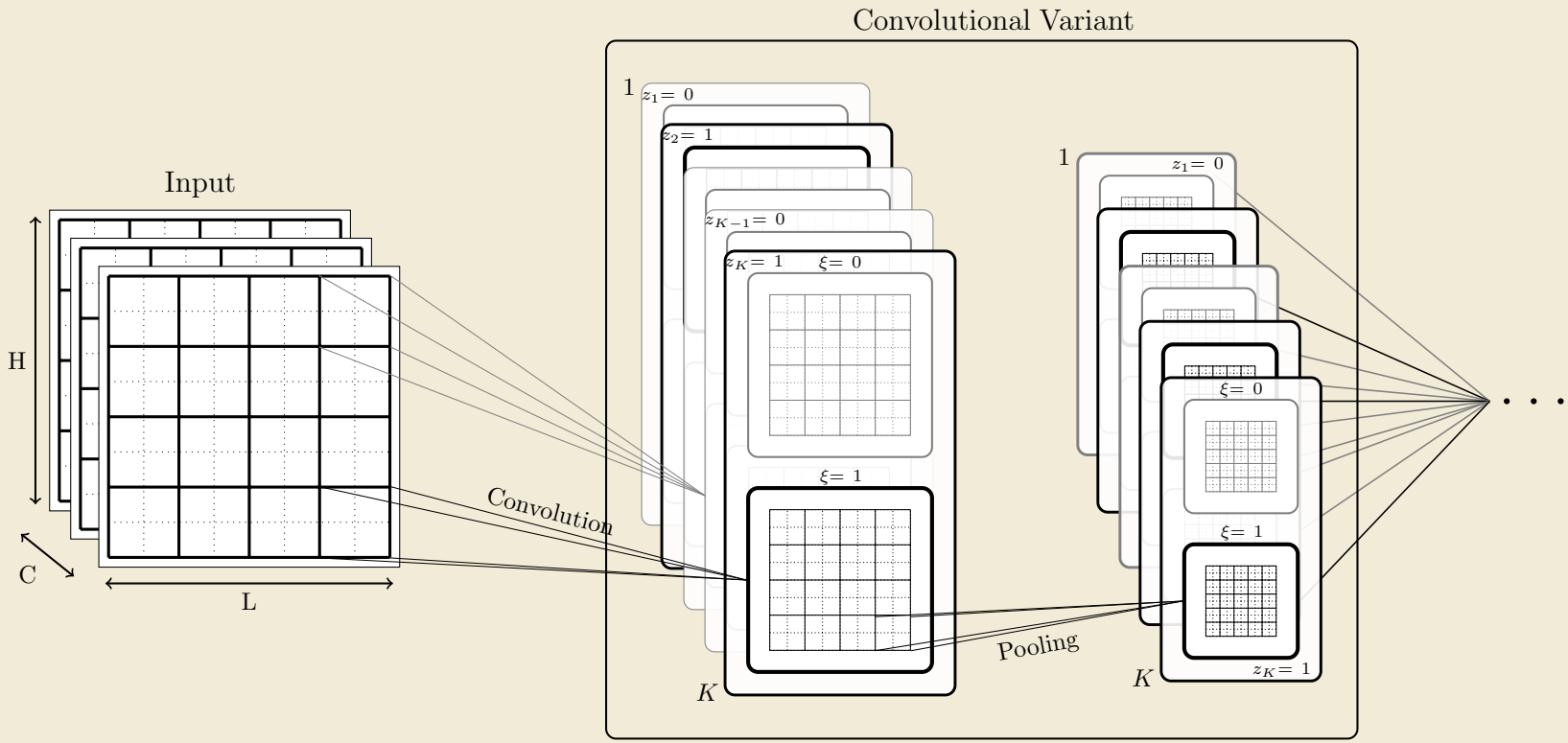
- After the previous reparameterization, the outputs $\hat{y}_{n,m},\ m|_1^M$, where M is the number of classes, are explicitly given as functions of the distribution defining (hyper)parameters, i.e.,

$$\hat{y}_{n,m} = \hat{y}_{n,m}\left(\mu_{j,k,u},\ \sigma^2_{j,k,u}, \tilde{\pi}_{j,k}, a_k, b_k\right),\ \forall\ j,k,u \text{ and all layers.}$$



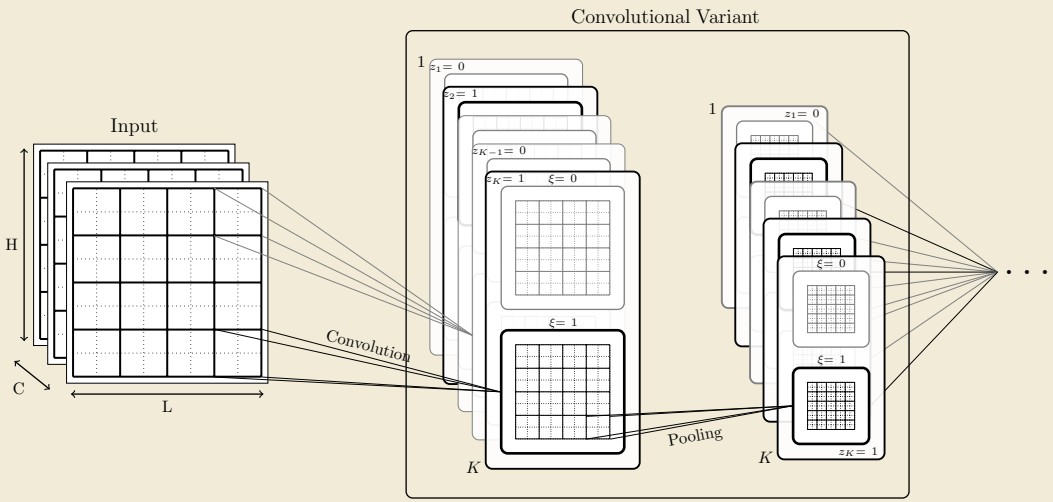
- The softmax nonlinearity is used for the output nodes. Thus, the outputs accept a probability interpretation.

The Nonparametric Bayesian Convolutional NN



In this case, the LWTAs consist of feature maps, i.e., images.

The Nonparametric Bayesian Convolutional NN



- Input volumes (tensors):
  $\{\boldsymbol{X}\}_{n=1}^{N} \in \mathbb{R}^{H \times L \times C}$
- The kernel weights:
  $\boldsymbol{W}_k \in \mathbb{R}^{h \times l \times C \times U},\ k|_1^K.$
- The utility hidden variables:
  $z_k \in \{0, 1\},\ k|_1^K.$
- The latent indicator variables:
  $\xi_{k,u},\ k|_1^K,\ u|_1^U,\ \boldsymbol{\xi}_k \in \mathbb{R}^U.$
- The output volumes per layer:
  $$\boldsymbol{Y}_{n,k} = \boldsymbol{\xi}_{n,k}\Big((\boldsymbol{W}_k \cdot z_k) \star \boldsymbol{X}_n\Big),$$
  $$\boldsymbol{Y}_{n,k} \in \mathbb{R}^{H \times L \times U},\ k|_1^K.$$

The Prior and Posterior Distributions of the RVs

- The distributions of all the rv's are as before. The difference is on how we obtain the softmax probabilities per LWTA unit.
- The posteriors: Categorical

$$q(\boldsymbol{\xi}_{n,k}) = \text{Cat}\left(\boldsymbol{\xi}\middle|P_{n,k,1},\ldots,P_{n,k,U}\right),\ k=1,2,\ldots$$

where the softmax nonlinearity has been computed as, i.e.,

$$P_{n,k,u} = \frac{\exp(h_{n,k,u})}{\sum_{u=1}^{U}\exp(h_{n,k,u})},$$

where in place of the responses, the sum of all the activations over the respective images for the U feature maps in the kth LWTA are employed, i.e.,

$$h_{n,k,u} = \sum_{h'=1}^{H}\sum_{l'=1}^{L}\Big[(z_k \cdot \boldsymbol{W}_{k,u}) \star \boldsymbol{X}_n\Big]_{h',l'}$$

The Cost Function

- Our goal is to learn the posterior distributions for a) the weights, b) the beta auxiliary variables and c) the utility variables. The latter ones are the variables that **prune** the network and set redundant units/kernels/weights to zero.
- Recall that the above posteriors have the form

$$\begin{aligned} q(w_{j,k,u}) &= \mathcal{N}\left(w|\mu_{j,k,u}, \sigma^2_{j,k,u}\right), \\ q(z_{j,k}) &= \text{Bern}\left(z|\tilde{\pi}_{j,k}\right), \\ q(\beta_k) &= \text{Beta}(\beta|a_k, b_k). \end{aligned}$$

- The parameters that have to be learned are the "red" variables that define the above posteriors.
- To learn parameters in distributions in the presence of hidden/latent variables, the expectation-maximization (EM) algorithm is mobilized.

The Cost Function

- Taken the specific forms of the priors and the conditionals in the current context, computations are not tractable.
- Thus, one has to resort to the variational approximation EM. To this end:
  - The mean field approximation of the posteriors is adopted, i.e.,

$$q(W, Z, \Xi, \boldsymbol{\beta}) = q_w(W) q_z(Z) q_\xi(\Xi) q_\beta(\boldsymbol{\beta}).$$

  - Instead of the likelihood, a lower bound, known as evidence lower bound (ELBO) of it is maximized, with respect to the unknown posteriors. The ELBO is given by

$$\ln p(\mathcal{D}; \Theta) \geq \mathbb{E}_q \left[\ln \frac{p(\mathcal{D}, W, Z, \Xi, \boldsymbol{\beta}; \Theta)}{q(W, Z, \Xi, \boldsymbol{\beta})} \right]$$

    where $\mathcal{D}$ denotes the set of the observations, $\mathcal{D} = \{(\boldsymbol{y}_n^o, \boldsymbol{x}_n)\}\big|_1^N$, and Θ all the unknown hyper-parameters.

The Cost Function

- Taking into account the specific prior distribution dependencies as well as the independence implied by the product of the involved posteriors, the cost can be rewritten as (notational dependence on Θ has been suppressed) :

$$\begin{aligned} \text{ELBO} \quad = \quad & \mathbb{E}_q\Big[\ln p(\mathcal{D}|W,Z,\Xi,\boldsymbol{\beta})\Big] + \mathbb{E}_{q_z,q_\beta} \ln \frac{p(Z|\boldsymbol{\beta})}{q_z(Z)} + \\ & \mathbb{E}_{q_\beta} \ln \frac{p(\boldsymbol{\beta})}{q_\beta(\boldsymbol{\beta})} + \mathbb{E}_{q_\xi} \ln \frac{p(\Xi)}{q_\xi(\Xi)} + \mathbb{E}_{q_w} \ln \frac{p(W)}{q_w(W)}, \end{aligned}$$

  where, $p(\cdot)$ are the priors and $q(\cdot)$ the posteriors, in their relaxed formulations, as they have been defined in the previous slides.

- Note that expectations are performed via sampling, using the reparametrized formulations. Usually, one sample draw is enough.

The Cost Function

- Note that all, but the first one, terms are the Kullback-Leibler divergences between the posteriors and the priors. This makes crystal clear the **regularizing** role of the priors.
- The first term is the conditional log-likelihood and it is computed by the output nodes,

$$\ln p(\mathcal{D}|W, Z, \Xi, \boldsymbol{\beta}) = \sum_{n=1}^{N} \sum_{m=1}^{M} y_{n,m}^{o} \ln \hat{y}_{n,m},$$

  where $\hat{y}_{n,m}$ are the softmax (probability) outputs, for each one of the output neurons; that is, the class **conditional** probabilities.
- Note that the above negative log-likelihood is the negative cross entropy cost function that is usually used as the loss function in training deep networks in the context of classification tasks.

The Cost Function

- Hence, the ELBO can be seen as a regularized version of the cross entropy loss function, averaged over a number of non-observable (hidden) variable samples, i.e.,

$$\begin{aligned}\text{ELBO} = & \underbrace{\mathbb{E}_q\big[\ln p(\mathcal{D}|W,Z,\Xi,\boldsymbol{\beta})\big]}_{\text{loss function}} + \\ & \underbrace{\mathbb{E}_{q_z,q_u}\ln\frac{p(Z|\boldsymbol{\beta})}{q_z(Z)}+\mathbb{E}_{q_\beta}\ln\frac{p(\boldsymbol{\beta})}{q_u(\boldsymbol{\beta})}}_{\text{regularizing terms}} + \\ & \underbrace{\mathbb{E}_{q_\xi}\ln\frac{p(\Xi)}{q_\xi(\Xi)}+\mathbb{E}_{q_w}\ln\frac{p(W)}{q_w(W)}}_{\text{regularizing terms}},\end{aligned}$$

- Note that in this context, the regularizing terms have a specific physical interpretation on each one of the hidden random variables that are used to build up a generative model.

Training and Inference

- For the optimization, standard stochastic gradient backpropagation arguments were followed, based on the ADAM algorithm.
- For the inference, we exploit the inferred component utility hidden variables. To this end, we introduce a cut-off threshold, τ; any component with inferred corresponding posterior, $q(z)$, below τ is removed from the network.

Quiz .....

- How many user-defined parameters (subjectivity) were needed to define the structure?

ONLY ONE

$$\text{Beta}(\beta|\alpha, 1)$$

Bit Precision

- The provision of a full Gaussian posterior distribution over the network weights, $w_{j,k,u}$, offers a natural way of reducing the floating-point **bit precision** level of the network implementation.
- The posterior variance of the network weights constitutes a measure of **uncertainty** in their estimates.
- We exploit this uncertainty information to assess which bits are **significant**, and remove the ones which fluctuate too much, under approximate posterior sampling.
- Note that in this way, the bit quantization accuracy is not arbitrary. It is learned from the data, via the optimization process and the resulting estimates of the respective Gaussian variances that are associated with the weights.

Experimental Results: LeNet-300-100 with MNIST.

- The goal: Compare LWTA nonlinearities regarding their classification performance and bit precision requirements, compared to using ReLU and Maxout. Two alternative configurations comprising: 1) 150 and 50 LWTA blocks on the first and second layer, respectively, of two competing units each, and 2) 75 and 25 LWTA blocks of four competing units. No pruning performed.

Activation	Error(%)	Bit Precision (Error %)
ReLU	1.60	2/4/10 (1.62)
Maxout/2 units	1.38	1/3/12 (1.57)
Maxout/4 units	1.67	2/5/12 (1.75)
SB-LWTA/2 units	1.31	1/3/11 (1.47)
SB-LWTA/4 units	1.34	1/2/8 (1.5)

Experimental Results: LeNet-300-100 with MNIST.

- The goal: Compare the proposed methods with previous ones with respect to a) pruning, b) classification performance and c) bit precision.

Method	Error (%)	# Weights	Bit precision
Original	1.6	$235K/30K/1K$	$23/23/23$
StructuredBP [Neklyudov]	1.7	$23,664/6,120/450$	$23/23/23$
Sparse-VD [Molchanov]	1.92	$58,368/8,208/720$	$8/11/14$
BC-GHS [Louizos]	1.8	$26,746/1,204/140$	$13/11/10$
SB-ReLU	1.75	$13.698/6.510/730$	$3/4/11$
SB-LWTA (2 units)	1.7	$12,522/6,114/534$	$2/3/11$
SB-LWTA (4 units)	1.75	$23,328/9,348/618$	$2/3/12$

Experimental Results: LeNet-5-Caffe with MNIST.

- The original LeNet-5-Caffe comprises a 5x5 kernel with 20 feature maps on the first layer, a 5x5 kernel with 50 feature maps on the second layer and a dense layer with 500 units on the third.
- In the convolutional SB-LWTA implementation, we consider 10 5x5 kernels (LWTA blocks) on the first layer, and 25 5x5 kernels on the second layer, with 2 competing feature maps each. We additionally consider an implementation comprising 4 competing feature maps with 5 5x5 kernels on the first layer, and 12 5x5 kernels on the second layer, reducing the total feature maps of the second layer to 48.

Experimental Results: LeNet-5-Caffe with MNIST.

Method	Error (%)	# Conv. Layers	Bit precision
Original	0.9	20/50	23/23/23/23
StructuredBP [Neklyudov]	0.86	3/18	23/23/23/23
Sparse-VD [Molchanov]	1.0	14/19	13/10/8/12
BC-GHS [Louizos]	1.0	5/10	10/10/14/13
SB-ReLU	0.9	10/16	8/3/3/11
SB-LWTA-2	0.9	6/6	6/3/3/13
SB-LWTA-4	0.8	8/12	11/4/1/11

Experimental Results: VGG on CIFAR-10.

- The VGG-like architecture comprises 13 convolutional layers, followed by a fully connected network of two layers.
- More specifically, the architecture consists of 3×3 kernels (same for all layers). The first two layers comprise 64 feature maps, followed by two layers with 128 feature maps, then three layers with 256 feature maps and six layers with 512 feature maps.
- The fully connected network comprises a single hidden with 512 nodes.

Experimental Results: VGG-like on CIFAR-10.

Method	Error(%)	r_w	Bit precision
BC-GNJ [Louizos]	8.6	6.57	10/10/10/10/8/8/8/5/5/5/5/5/6/7/11
BC-GHS [Louizos]	9.0	5.4	11/12/9/14/10/8/5/5/6/6/6/8/11/17/10
SB-LWTA-2	8.5	5.18	10/10/7/10/7/5/6/5/7/7/6/6/9/14/10

In the table, r_w is the ratio of non-zero weights after pruning.

The Possible WHYs Concerning the Results

- Recalling Hinton ([14]): "Supervised neural networks generalize well if there is **much less** information in the weights than there is in the output vectors of the training cases. So during learning, it is important to keep the weights simple by penalizing the **amount of information** they contain.
- The Bayesian perspective of pruning and reducing bit precision for the weights is aligned with achieving high accuracy. This is because Bayesian methods search for the optimal model structure and reward uncertain posteriors over parameters, according to Hinton's arguments. Bayesian Inference is directly related to the MDL (Rissanen's Minimum Description Length) principle.

The Possible WHYs Concerning the Results

- The use of Bayesian nonparametrics allows the direct incorporation of dedicated hidden (utility) variables that are concerned on the necessity or not of presence of nodes/kernels/weights. Their significance is not deduced indirectly from the weights. Utility variables are learned form the data, in parallel with the weights.
- Does the use of LTWA's help in the pruning? This is the evidence from the experiments. Further thought and experimentation is required .....

HAPPY (?) END

THANKS FOR YOUR PATIENCE

The Local Winner-Take-All Unit

[1] M. Denil, B. Shakibi, L. Dinh, N. de Freitas, et al. "Predicting parameters in deep learning" In Advances in Neural Information Processing Systems, pp. 2148–2156, 2013. [2] J. Bill and R. Legenstein "A compound memristive synapse model for statistical learning through STDP in spiking neural networks," Front. Neurosci., 8:41, (2014).
[3] Z. Yu, Y. Tian, T. Huang "Winner-Take-All as Basic Probabilistic Inference Unit of Neuronal Circuits", arXiv:1808.00675v1 [q-bio.NC] 2 Aug 2018.
[4] P. Anderson, G. N. Gross, T. Lømo, O. Sveen "Participation of inhibitory and excitatory interneurones in the control of hippocampal cortical output", In Mary A.B. Brazier, editor, The Interneuron, volume 11, University of California Press, Los Angeles, 1969.
[5] J. C. Eccles, M. Ito, J. Szentágothai "The cerebellum as a neuronal machine", Springer-Verlag New York, 1967.
[6] C. Stefanis "Interneuronal mechanisms in the cortex", In Mary A.B. Brazier, editor, The Interneuron, volume 11, University of California Press, Los Angeles, 1969.
[7] W. Maass "On the computational power of winner-take-all", Neural Computation, 12:2519–2535, 2000.
[8] C. J. Maddison, A. Mnih, Y. W. Teh,"The concrete distribution: A continuous relaxation of discrete random variables", In Proc. ICLR, 2017.
[9] P. Kumaraswamy, "A generalized probability density function for double-bounded random processes", Journal of Hydrology, Vol. 46 (1), pp. 79-88, 1980.
[10] E. Nalisnick, P. Smyth "Stick-breaking variational autoencoders", Proc. ICLR, 2016.
[11] C. Louizos, K. Ullrich, M. Welling, "Bayesian compression for deep learning". In Proc. NIPS, pp. 3290-3300, 2017.
[12] D. Molchanov, A. Ashukha, D. Vetrov, "Variational dropout sparsifies deep neural networks", In Proc. ICML 34, pp. 2498–2507, 2017.
[13] K. Neklyudov, D. Molchanov, A. Ashukha, D. Vetrov, "Structured Bayesian pruning via log-normal multiplicative noise", In Proc. NIPS 31, pp. 6775–6784. 2017.
[14] G. E. Hinton, D. Van Camp, "Keeping the neural networks simple by minimizing the description length of the weights," In Proceedings of the sixth annual conference on Computational learning theory, pp. 5–13. ACM, 1993.
[16] K. Panousis, S. Chatzis, S. Theodoridis "Nonparametric Bayesian Deep Networks with Local Competition", Proceedings ICML, 2019.