SNR（Sub-Network-Routing)

出自论文 SNR: Sub-Network Routing for Flexible Parameter Sharing in Multi-Task Learning。MMoE主要针对多个共享的expert 网络的输出进行attention组合(也就是门控)。SNR 在这种模块化的基础上，使用编码变量（coding variables）控制子网络之间的连接，实现多任务模型中不同程度的参数共享。

SNR 的提出主要解决级间的参数共享问题，达到最佳组合的网络结构。简言之，SNR和MMoE的不同之处就是，MMoE拿多个子网络的输出做加权直接输入到了每个任务各自的tower中；而SNR对不同子网络的输出进行组合又输入到了下一层子网络，形成子网络的组合。

这种方法比MMoE更加灵活，因为通过coding variable（0/1），我们可以控制模块和模块之间的连接关系，这就是网络结构搜索（NAS）的思想。例如看下图中间的那个图，如果某些虚线是0的话（即coding variable = 0），就代表网络没有连接，那么某个task的网络结构就发生改变了，这也正是我们要的效果 -- 不同任务的网络结构是不同的，相当于对不同任务进行NAS。而MMoE中每个task的网络结构是相同的，只不过使用了不同的attention net。

coding variable的核心意义在于引入sparsity，即当它为0的时候实现了网络结构的选择，而且计算一个task的参数量就变小了。"allows sparse connections among sub-networks, which further improves the serving computational efficiency."

SNR设计了两种类型的连接方式：SNR-Trans 和 SNR-Aver来学习**子网络的组合**，最终得到特定多任务场景的**最优网络结构**。SNR-Trans和SNR-Aver的区别在于，SNR-Trans模块和模块之间的连接是矩阵；SNR-Aver模块和模块之间的连接是标量。

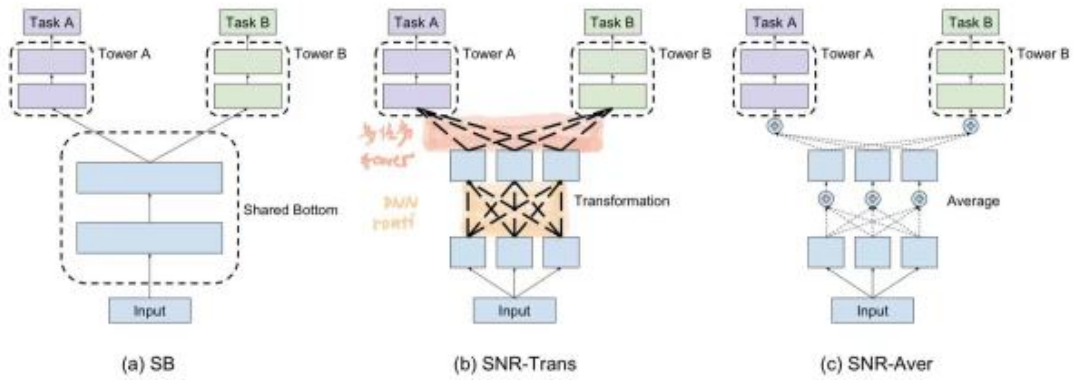

Figure 1: Multi-task Models. (a) Shared-Bottom (SB) model: the conventional multi-task neural network model. (b) Sub-Network Routing with Transformation (SNR-Trans) model: the shared layers are split into sub-networks and the connection (dashed line) between the sub-networks is a transformation matrix multiplied by a scalar latent variable. (c) Sub-Network Routing with Average (SNR-Aver) model: the shared layers are split into sub-networks and the connection (dashed line) between the sub-networks is a weighted average with scalar latent variables as weights.

- SNR−Trans模型：将共享层划分为子网络，子网络之间的连接(虚线)为变换矩阵W乘以coding variable z:

$$\begin{bmatrix} \boldsymbol{v_1} \\ \boldsymbol{v_2} \end{bmatrix} = \begin{bmatrix} z_{11}\boldsymbol{W_{11}} & z_{12}\boldsymbol{W_{12}} & z_{13}\boldsymbol{W_{13}} \\ z_{21}\boldsymbol{W_{21}} & z_{22}\boldsymbol{W_{22}} & z_{23}\boldsymbol{W_{23}} \end{bmatrix} \begin{bmatrix} \boldsymbol{u_1} \\ \boldsymbol{u_2} \\ \boldsymbol{u_3} \end{bmatrix}$$

其中，$\boldsymbol{u_1}, \boldsymbol{u_2}, \boldsymbol{u_3}$ 代表low-level sub-network的输出；$\boldsymbol{v_1}, \boldsymbol{v_2}$ 代表high-level sub-network的输入；$\boldsymbol{W_{i,j}}$代表连接 $\boldsymbol{j}$low-level到 $\boldsymbol{i}$high-level的权重矩阵，$\boldsymbol{z_{i,j}}$代表 $\boldsymbol{j}$low-level到 $\boldsymbol{i}$ high-level的连接性coding variable（二元变量，0 or 1）。如：$\mathbf{v}_1 = z_{11}\mathbf{W}_{11}\mathbf{u}_1 + z_{12}\mathbf{W}_{12}\mathbf{u}_2 + z_{13}\mathbf{W}_{13}\mathbf{u}_3$.

可以想见，如果所有的连接参数z都是1的话，整个网络退化成shared-bottom；如果只有z_{11}和z_{22} = 1，其他都是0的话，那么整个网络退化成两个独立的任务。总之，我们就是要通过学习coding variable，来得到**两个task最优的网络结构。**

$\boldsymbol{W}$和$\boldsymbol{z}$是我们要学习的参数，和NAS一样，我们要同时去学习网络参数和网络结构。假设$\boldsymbol{z}$ 服从 Bernoulli 分布，用0/1来控制网络的连接与断开。但是Bernoulli分布是不可微的，因此需要把$\boldsymbol{z}$做一个变换，变换成一个平滑的函数:

$$u \sim U(0,1), s = \text{sigmoid}((\log(u) - \log(1-u) + \log(\alpha)/\beta)$$
$$\bar{s} = s(\zeta - \gamma) + \gamma, z = \min(1, \max(\bar{s}, 0)),$$

where u is a uniform random variable, $\log(\alpha)$ is a learnable distribution parameter, and β, γ, ζ are all hyper-parameters.

可以看到，当$\bar{s} < 0$时，有 z = 0，即此时起到了网络结构选择的作用!

- SNR−Aver模型：和SNR-Trans相比，就是中间连接不用矩阵了，而是变成了简单的求和。coding variable依然是起到**网络结构选择**的作用，毕竟我们希望不同的task对应的网络结构都不一样。在同样的参数budget的条件下，由于SNR-Aver少了矩阵连接的参数，所以可以把更多的参数分配到底下的shared 子网络中。

$$\begin{bmatrix} \boldsymbol{v_1} \\ \boldsymbol{v_2} \end{bmatrix} = \begin{bmatrix} z_{11}\boldsymbol{I_{11}} & z_{12}\boldsymbol{I_{12}} & z_{13}\boldsymbol{I_{13}} \\ z_{21}\boldsymbol{I_{21}} & z_{22}\boldsymbol{I_{22}} & z_{23}\boldsymbol{I_{23}} \end{bmatrix} \begin{bmatrix} \boldsymbol{u_1} \\ \boldsymbol{u_2} \\ \boldsymbol{u_3} \end{bmatrix}$$