Highway Networks

(R. K. Srivastava, Greff, and Schmidhuber 2015)

Standard NN layer: $y = H(x, W_H)$ where H is a non-linear transformation parametrized by weights W_H

Highway network:

$$y = H(x, W_H) \cdot T(x, W_T) + x \cdot C(x, W_C)$$

where T is the *transform* gate and C is the *carry* gate, which define the ratio in which the output is defined by transforming the input in contrast to carrying it over. For simplicity, $C = 1 - T$, producing:

$$y = H(x, W_H) \cdot T(x, W_T) + x \cdot (1 - T(x, W_T))$$

Note: this formulation, where each layer can propagate its input x further requires that all of the elements have the same dimension (y, x, T, H). An option here is to use padding to upscale x or sub-sampling, in order to reduce the dimensionality. An option is also to use a regular layer (without the highway connections) to change the dimensionality, and then continue with the highway layers.

Recurrent Highway Networks

(Zilly et al. 2016)

The paper sketches out the proof of the vanishing/exploding gradient problem and isolates its relation to the largest singular value of the recurrent weight matrix:

$$|A| \leq \left|R^T\right| \left|diag[f'(Ry^{[t-1]})]\right| \leq \gamma\sigma_{max}$$

The gradient vanishes when $\gamma\sigma_{max} < 1$ and explodes when the expression is larger than one.

γ is the maximum value the gradient of the activation function.

Geršgorin circle theorem states:

$$spec(A) \subset \bigcup_{i \in \{1,\ldots,n\}} \left\{ \lambda \in \mathbb{C} \left\| \lambda - a_{ii} \right\|_{\mathbb{C}} \leq \sum_{j=1, j \neq i}^{n} |a_{ij}| \right\}$$

translated, the spectrum of eigenvelues of the square matrix $A \in \mathbb{R}^{n \times n}$ lies within the union of complex circles which are centered around the **diagonal values** of the matrix A, with a radius equal to the sum of the absolute values of the **non-diagonal** entries of each row.

Essentially, this means that shifting the diagonal values shifts the center of the circles, and therefore the possible location of the eigenvalues. Also, increasing the values of the remaining elements increases the radius in which the eigenvalues are contained.

TODO: Add viz of theorem w/TikZ

Initialization of recurrent weights as mentioned in (Le, Jaitly, and Hinton 2015), one way to circumvent this via initialization is to initialize the recurrent matrix as an identity matrix and the remainder as small random values. However, this method does nothing to mitigate the fact that the values of the matrix will change during training, resulting in the same exploding / vanishing gradient phenomenon.

Essentially, a reformulation of a RNN in the form of a vertical highway network is used (more or less equal to LSTM, where the previous cell state is propagated input).

Takeaways:

- most of the transform-processing is done in the first layer, and then to a lesser extent (first layer contextually transforms the input features? and the remaining layers use this contextual information).
- passing the input along in a resnet-like or highway-like fashion is useful.
- Geršgorin can help limit the range of singular values of a matrix.

Learning long term dependencies Dataset: pixel-by-pixel MNIST image classification, introduced in (Le, Jaitly, and Hinton 2015)

Learning long-term dependencies in RNNs with Auxilliary Losses

(Trinh et al. 2018)

- Randomly sample one or multiple anchor positions
- Use an unsupervised auxilliary loss
- Reconstruction auxilliary loss (reconstruct a subsequence given first symbol, enhances remembering)
- Prediction auxilliary loss (predict a future token in language-model fashion)
- Trained in two phases:
- Pure unsupervised pretraining (minimize auxilliary loss)
- Semi-supervised learning where $min_\theta L_{sup}(\theta) + L_{aux}(\theta')$

Hyperparameters: - How frequently to sample the reconstructon segments, and how long they are

The methods help with learning long-term dependencies, even when the backprop is truncated. Essentially, signal is needed for the network to remember things. That signal can't be achieved through very long backprop due to failure of credit assignment.

Additional ablation study & result analysis in paper.

Dropout

Introduced in: (Hinton et al. 2012)

Backpropagation

Introduced in: (Rumelhart, Hinton, and Williams 1985)

Maxout networks

(Goodfellow et al. 2013)

Maxout networks use the *maxout* function as the activation. For an input $x \in \mathbb{R}^d$ the maxout is:

$$h_i(x) = max_{j \in [1,k]} z_{ij}$$

where $z_{ij} = x^T W_{...ij} + b_{ij}$, $W \in \mathbb{R}^{d \times m \times k}$ and $b \in \mathbb{R}^{m \times k}$ are the learned model parameters.

Essentially: instead of projecting into the output dimension m, project into $m \times k$ and max over the k additional dimensions. Pytorch impl: link

Grid LSTM

(Kalchbrenner, Danihelka, and Graves 2015)

Similar to Multi-dimensional Recurrent Neural Networks (Graves and Schmidhuber 2009)

LSTM along each dimension of network (depth, T). The vertical LSTM hidden / cell states initialized by the inputs.

N-dimensional Grid LSTM accepts N hidden vectors $h_1, \ldots, h_N$ and N memory vectors $m_1, \ldots, m_N$, which are all distinct for each dimension.

All of the hidden states are then concatenated:

$$H = \begin{bmatrix} \hat{h}_1 \\ \vdots \\ \hat{h}_N \end{bmatrix} \tag{1}$$

The N-dimensional block then computes N LSTM tranfsorms, one for each dimension. Each LSTM transform has its individual weight matrices. Each block accepts input hidden and memory vectors from N dimensions, and outputs them into N dimensions.

$$\begin{aligned} (\hat{h}_1, \hat{m}_1) &= LSTM(H, m_1, W_1) \\ &\ldots \\ (\hat{h}_N, \hat{m}_N) &= LSTM(H, m_N, W_N) \end{aligned} \tag{2}$$

CONT

Learning to Skim Text

(Yu, Lee, and Le 2017) : LSTM-Jump

- Uses a policy gradient method to make decisions to skip a number of tokens in the input
- Hyperparam: max jump size K, number of tokens to read before jumping R
- Processing stops if
  - Jumping softmax predicts 0
  - Jump exceeds sequence length N
  - The network processed all tokens
- The last hidden state is used for prediction in downstream tasks

REINFORCE procedure (+Baselines):

Objective:

- Minimize cross-entropy error (classification loss)
- Maximize expected reward under the current jumping policy (R = -1 for misclassification, +1 for correct)
- Baselines regularization term: minimize difference between actual reward and predicted baseline

Neural Speed Reading via Skim-RNN

(Seo et al. 2017)

OpenReview discussion: link

Update just a part of the hidden state for irrelevant words (uses a smaller RNN)

The hard decision (whether the word is important or not) isn't differentiable – the gradient is approximated by Gumbel-Softmax instead of REINFORCE (policy gradient). The method results in a reduced number of FLOPs (floating point operations).

"skimming achieves higher accuracy compared to skipping the tokens, implying that paying some attention to unimportant tokens is better than completely ignoring (skipping) them"

RNNs with hard decisions – last paragraph of chapter 2, references

Model:

- Two RNNs, default (big), skim (small)
- Binary hard decision of skimming is a stochastic multinomial variable over the probabilities of a single layer NN which accepts the next token and current hidden state
- During inference, instead of sampling, the greedy argmax of the multinomial parameters is used

$$h_t = \begin{cases} f(x_t, h_{t-1}), & \text{if } Q_t = 1 \\ [f'(x_t, h_{t-1}); h_{t-1}[d'+1:d]], & \text{if } Q_t = 2 \end{cases}$$

where $Q_t = 1$ means the network chose to fully read and $Q_t = 2$ means the network has decided to skim.The dimension k of the multinomial distribution is 2.

Gumbel-softmax:

Expected loss over the sequence of skim-read decisions is:

$$\mathbb{E}_{Q_t \sim Multinomial(p_t)}\left[L(\sigma)\right] = \sum_Q L(\sigma; Q)P(Q) = \sum_Q L(\sigma; Q) \prod_j p_j^{Q_j}$$

to approximate the expected loss, all of the hard decisions Q_t need to be enumerated and evaluated, which is intractable. One approximation of the gradient is by using REINFORCE which while unbiased has high variance. A replacemnt is to use the gumbel-softmax distribution (Jang, Gu, and Poole 2016).

The reparametrization constructs a softmax over the probabilities of the multinomial distribution with an added Gumbel noise:

$$r_t^i = \frac{exp((log(p_t^i) + g_t^i)/\tau)}{\sum_j exp((log(p_t^j) + g_t^j)/\tau)}$$

where τ is a temperature hyperparameter, and g_t^i is an independent sample from $Gumbel(0,1) = -log(-log(Uniform(0,1)))$. There are two distinct r^i's, one for skim and one for fully read.

To encourage the model to *skim when possible*, a regularization term is added which minimizes the mean of the negative log probability of skimming $\frac{1}{T}\sum log(p_t^2)$:

$$L'(\sigma) = L(\sigma) + \gamma \frac{1}{T} \sum -log(p_t^2)$$

Discussion on openreview:

Similar to: (Jernite et al. 2016)

Variable Computation in Recurrent Neural Networks

(Jernite et al. 2016)

Variable Computation GRU and Variable Computation RNN (VCGRU, VCRNN)

At each timestep t, the *scheduler* takes the current hidden and input vectors and decides on the number of dimensions to use for the update (d). The first d dimensions of the **hidden state and input vector (!)** are then used to compute the first d elements of the new hidden state, while the rest is carried over from the previus state.

Scheduler: function $m : \mathbb{R}^{2D} \rightarrow [0, 1]$ decides which portion of the hidden state to change. For each timestep t:

$$m_t = \sigma(u \cdot h_{t-1} + v \cdot x_t + b)$$

The first $\lceil m_t D \rceil$ dimensions are then the ones updated. In the lower-dimensional recurrent unit, the upper left sub-square matrices of shape $d \times d$ are used.

Soft masking: the decision to update only a subset of a state is essentially a hard chouce and makes the model non-differentianle. The hard choice is approximated by using a gating function which applies a soft mask.

The gating vector e_t is define by:

$$\forall i \in 1, \ldots, D, (e_t)_i = \text{Thres}_\epsilon(\sigma(\lambda(m_t D - i)))$$

Where λ is a *sharpness parameter*, and Thres maps all values greater than $1 - \epsilon$ and smaller than ϵ to 1 and 0.

Annotation Artifacts in Natural Language Inference Data

(Gururangan et al. 2018)

Classification model on just the *hypothesis* of NLI achieves 67% on SNLI and 53% on MultiNLI.

"Negation and vagueness are highly correlated with ceratin inference classes. Our findings suggest that the success of natural language inference models to date has been overestimated, and that the task remains a hard open problem." ->

- entailed hypotheses contain gender-neutral references to people
- purpose clauses are a sign of neutral hypotheses
- negation correlates with contradiction

Annotation artifacts: patterns in the data that occur as a result of the framing of the annotation task influencing the language used by the crowd workers.

Discussion: "Many datasets contain annotation artifacts..." – references to other examples of this phenomenon

References

Goodfellow, Ian J, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. 2013. "Maxout Networks." *arXiv Preprint arXiv:1302.4389*.

Graves, Alex, and Jürgen Schmidhuber. 2009. "Offline Handwriting Recognition with Multidimensional Recurrent Neural Networks." In *Advances in Neural Information Processing Systems*, 545–52.

Gururangan, Suchin, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel R Bowman, and Noah A Smith. 2018. "Annotation Artifacts in Natural Language Inference Data." *arXiv Preprint arXiv:1803.02324*.

Hinton, Geoffrey E, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. 2012. "Improving Neural Networks by Preventing

Co-Adaptation of Feature Detectors." *arXiv Preprint arXiv:1207.0580*.

Jang, Eric, Shixiang Gu, and Ben Poole. 2016. "Categorical Reparameterization with Gumbel-Softmax." *arXiv Preprint arXiv:1611.01144*.

Jernite, Yacine, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. "Variable Computation in Recurrent Neural Networks." *arXiv Preprint arXiv:1611.06188*.

Kalchbrenner, Nal, Ivo Danihelka, and Alex Graves. 2015. "Grid Long Short-Term Memory." *arXiv Preprint arXiv:1507.01526*.

Le, Quoc V, Navdeep Jaitly, and Geoffrey E Hinton. 2015. "A Simple Way to Initialize Recurrent Networks of Rectified Linear Units." *arXiv Preprint arXiv:1504.00941*.

Rumelhart, David E, Geoffrey E Hinton, and Ronald J Williams. 1985. "Learning Internal Representations by Error Propagation." California Univ San Diego La Jolla Inst for Cognitive Science.

Seo, Minjoon, Sewon Min, Ali Farhadi, and Hannaneh Hajishirzi. 2017. "Neural Speed Reading via Skim-Rnn." *arXiv Preprint arXiv:1711.02085*.

Srivastava, Rupesh Kumar, Klaus Greff, and Jürgen Schmidhuber. 2015. "Highway Networks." *arXiv Preprint arXiv:1505.00387*.

Trinh, Trieu H, Andrew M Dai, Minh-Thang Luong, and Quoc V Le. 2018. "Learning Longer-Term Dependencies in Rnns with Auxiliary Losses."

Yu, Adams Wei, Hongrae Lee, and Quoc V Le. 2017. "Learning to Skim Text." *arXiv Preprint arXiv:1704.06877*.

Zilly, Julian Georg, Rupesh Kumar Srivastava, Jan Koutník, and Jürgen Schmidhuber. 2016. "Recurrent Highway Networks." *arXiv Preprint arXiv:1607.03474*.