The DROPOUT Regularization Technique (Hinton et al., 2012)

Dropout is a data-driven regularization technique for neural nets. Goal = prevent feature detectors to rely on each other.
For layers to which dropout is applied, for each training example:
☆ At training time randomly drop neurons with probability $p$ = simultaneously train many networks that share weights
☆ At test time, keep all neurons, scale outgoing weights by $(1 - p)$ = use the full capacity and combine all subarchitectures

**Task**
Handwritten Text Line Recognition

**Rimes**
- Training: 10,203 lines / 460k characters (97 unique)
- Test: 1,130 lines / 52k characters
- Language Model: 5k-word vocabulary + 4-gram (2.9% OOV; perplexity 18)

**IAM**
- Training: 6,482 lines / 288k characters (79 unique)
- Test: 976 lines / 43k characters
- Language Model: 50k-word vocabulary + 3-gram (4.3% OOV; perplexity 298)

**Bentham**
- Training: 9,198 lines / 420k characters (93 unique)
- Test: 1,415 lines / 64k characters
- Language Model: 32k-word vocabulary + 3-gram (5.6% OOV; perplexity 103)

**Success of dropout**
Deep MLPs, ImageNet (Hinton et al., 2012)
ConvNets (Deng et al., 2013)
Recurrent nets (Mesnil et al., 2013)
LSTM nets (Pham et al., 2014, Zaremba et al. 2014)
Equivalent to L2 regularization (Wager et al., 2013)

**Model: BLSTM-RNN**
Bidirectional Long Short-Term Memory Recurrent Neural Network

**Inputs**
- Handcrafted features (Blame-Bernard, 2011)
- or raw pixel values, extracted with a sliding window

**Outputs**
- one class per character
- one non-character (blank) class

**Training**
Connectionist Temporal Classification (CTC; Graves et al., 2006) with SGD and BPTT = train to predict character sequences from whole text lines

**Effect of the POSITION of Dropout**
The position of dropout has an effect on the relative character error rate (CER) improvement (over no regularization)

Averaged over all six configurations, dropout seems more beneficial:
- at the bottom layer of the RNN
- before the LSTM units

**Effect on the WEIGHTS**
With dropout on the inputs, the local correlations of neighboring pixels are less visible to the network during training.
With half the pixels missing, the model cannot rely on regularities in the input signal.

**Results of RNN ALONE (Char.Err.Rate; CER%)**

\[ c^* = \arg \max_{c} \text{Pr}(c|x) \]

- dropout is always helpful regardless of the relative position to the LSTMs (compared to no regularization)
- before the LSTMs is better than after the LSTMs (improvement over Pham et al. (2012))

(dropout is applied in all layers)

**Results of RNN+LM (Word.Err.Rate; WER%)**

\[ c^* = \arg \max \sum_{c} \left( \frac{\text{Pr}(c|x)}{\text{Pr}(c|x) + \text{Pr}(c|y)} \right) \cdot \text{Pr}(y|x) \]

- dropout is not always helpful
- In half the configurations, it was better after the LSTMs than before
- In the complete pipeline, regularization of the classification weights (after the top LSTM) seems important!
- Best results with combination of before lower LSTMs and after top LSTM

**Conclusions**
The different experiments showed that it is difficult to draw a general conclusion about the best position.
Applying it only after LSTM layers (Pham et al., 2014), or between every layer (Zaremba et al. 2014), is not always optimal.
Before LSTMs seems to be a good choice, as well as after the top LSTM (classification weights) for complete systems

{tb,jl}@a2ia.com