Gru & Attention

Manfred Klenner

Department of Computerlinguistik
University of Zurich

November 9, 2020

Vanishing and exploding gradients

- if we use gradient-based learning (descend) and backpropagation
- given deep architectures the chain rule produces a lot of products
- if gradients are small, multiplication makes it ever smaller
- weights then might no longer change
- may completely stop the neural network from further training
- in this worst case, the neural network might became static

Vanishing and exploding gradients

- if we use gradient-based learning (descend) and backpropagation
- given deep architectures the chain rule produces a lot of products
- if gradients are small, multiplication makes it ever smaller
- weights then might no longer change
- may completely stop the neural network from further training
- in this worst case, the neural network might became static

Vanishing and exploding gradients

- if we use gradient-based learning (descend) and backpropagation
- given deep architectures the chain rule produces a lot of products
- if gradients are small, multiplication makes it ever smaller
- weights then might no longer change
- may completely stop the neural network from further training
- in this worst case, the neural network might became static

Vanishing and exploding gradients

- if we use gradient-based learning (descend) and backpropagation
- given deep architectures the chain rule produces a lot of products
- if gradients are small, multiplication makes it ever smaller
- weights then might no longer change
- may completely stop the neural network from further training
- in this worst case, the neural network might became static

Vanishing and exploding gradients

- if we use gradient-based learning (descend) and backpropagation
- given deep architectures the chain rule produces a lot of products
- if gradients are small, multiplication makes it ever smaller
- weights then might no longer change
- may completely stop the neural network from further training
- in this worst case, the neural network might became static

Vanishing and exploding gradients

- if we use gradient-based learning (descend) and backpropagation
- given deep architectures the chain rule produces a lot of products
- if gradients are small, multiplication makes it ever smaller
- weights then might no longer change
- may completely stop the neural network from further training
- in this worst case, the neural network might became static

Vanishing and exploding gradients: ways to go

- residual (skip) connections
  - they allow gradient information to pass through the layers
  - creating "highways" of information
  - output of a previous layer is added to the output of a deeper layer
- change activation function
  - e.g ReLU only saturates in one direction (but now exploding is possible)
- use LSTM, GRU to prevent vanishing
  - the forget gate gets rid of some values
  - the cell states is used additively (preventing values from getting smaller)
  - we only have positives values, since we are using sigmoid function
- use gradient clipping to prevent exploding
  - use a threshold on the gradients
  - if $||g|| > \text{threshold}$ $g = \frac{threshold * g}{||g||}$ where $||.||$ denotes the length
- weight regularization: L1 or L2

Vanishing and exploding gradients: ways to go

- residual (skip) connections
  - they allow gradient information to pass through the layers
  - creating "highways" of information
  - output of a previous layer is added to the output of a deeper layer
- change activation function
  - e.g ReLU only saturates in one direction (but now exploding is possible)
- use LSTM, GRU to prevent vanishing
  - the forget gate gets rid of some values
  - the cell states is used additively (preventing values from getting smaller)
  - we only have positives values, since we are using sigmoid function
- use gradient clipping to prevent exploding
  - use a threshold on the gradients
  - if $||g|| > \text{threshold}$ $g = \frac{threshold * g}{||g||}$ where $||.||$ denotes the length
- weight regularization: L1 or L2

Vanishing and exploding gradients: ways to go

- residual (skip) connections
  - they allow gradient information to pass through the layers
  - creating "highways" of information
  - output of a previous layer is added to the output of a deeper layer
- change activation function
  - e.g ReLU only saturates in one direction (but now exploding is possible)
- use LSTM, GRU to prevent vanishing
  - the forget gate gets rid of some values
  - the cell states is used additively (preventing values from getting smaller)
  - we only have positives values, since we are using sigmoid function
- use gradient clipping to prevent exploding
  - use a threshold on the gradients
  - if $||g|| > \text{threshold}$ $g = \frac{threshold * g}{||g||}$ where $||.||$ denotes the length
- weight regularization: L1 or L2

Vanishing and exploding gradients: ways to go

- residual (skip) connections
  - they allow gradient information to pass through the layers
  - creating "highways" of information
  - output of a previous layer is added to the output of a deeper layer
- change activation function
  - e.g ReLU only saturates in one direction (but now exploding is possible)
- use LSTM, GRU to prevent vanishing
  - the forget gate gets rid of some values
  - the cell states is used additively (preventing values from getting smaller)
  - we only have positives values, since we are using sigmoid function
- use gradient clipping to prevent exploding
  - use a threshold on the gradients
  - if $||g|| > \text{threshold}$ $g = \frac{threshold * g}{||g||}$ where $||.||$ denotes the length
- weight regularization: L1 or L2

Vanishing and exploding gradients: ways to go

- residual (skip) connections
  - they allow gradient information to pass through the layers
  - creating "highways" of information
  - output of a previous layer is added to the output of a deeper layer
- change activation function
  - e.g ReLU only saturates in one direction (but now exploding is possible)
- use LSTM, GRU to prevent vanishing
  - the forget gate gets rid of some values
  - the cell states is used additively (preventing values from getting smaller)
  - we only have positives values, since we are using sigmoid function
- use gradient clipping to prevent exploding
  - use a threshold on the gradients
  - if $||g|| > \text{threshold}$ $g = \frac{threshold * g}{||g||}$ where $||.||$ denotes the length
- weight regularization: L1 or L2

Vanishing and exploding gradients: ways to go

- residual (skip) connections
  - they allow gradient information to pass through the layers
  - creating "highways" of information
  - output of a previous layer is added to the output of a deeper layer
- change activation function
  - e.g ReLU only saturates in one direction (but now exploding is possible)
- use LSTM, GRU to prevent vanishing
  - the forget gate gets rid of some values
  - the cell states is used additively (preventing values from getting smaller)
  - we only have positives values, since we are using sigmoid function
- use gradient clipping to prevent exploding
  - use a threshold on the gradients
  - if $||g|| > \text{threshold}$ $g = \frac{threshold * g}{||g||}$ where $||.||$ denotes the length
- weight regularization: L1 or L2

Vanishing and exploding gradients: ways to go

- residual (skip) connections
  - they allow gradient information to pass through the layers
  - creating "highways" of information
  - output of a previous layer is added to the output of a deeper layer
- change activation function
  - e.g ReLU only saturates in one direction (but now exploding is possible)
- use LSTM, GRU to prevent vanishing
  - the forget gate gets rid of some values
  - the cell states is used additively (preventing values from getting smaller)
  - we only have positives values, since we are using sigmoid function
- use gradient clipping to prevent exploding
  - use a threshold on the gradients
  - if $||g|| > \text{threshold}$ $g = \frac{threshold * g}{||g||}$ where $||.||$ denotes the length
- weight regularization: L1 or L2

Vanishing and exploding gradients: ways to go

- residual (skip) connections
  - they allow gradient information to pass through the layers
  - creating "highways" of information
  - output of a previous layer is added to the output of a deeper layer
- change activation function
  - e.g ReLU only saturates in one direction (but now exploding is possible)
- use LSTM, GRU to prevent vanishing
  - the forget gate gets rid of some values
  - the cell states is used additively (preventing values from getting smaller)
  - we only have positives values, since we are using sigmoid function
- use gradient clipping to prevent exploding
  - use a threshold on the gradients
  - if $||g|| > \text{threshold}$ $g = \frac{threshold * g}{||g||}$ where $||.||$ denotes the length
- weight regularization: L1 or L2

Vanishing and exploding gradients: ways to go

- residual (skip) connections
  - they allow gradient information to pass through the layers
  - creating "highways" of information
  - output of a previous layer is added to the output of a deeper layer
- change activation function
  - e.g ReLU only saturates in one direction (but now exploding is possible)
- use LSTM, GRU to prevent vanishing
  - the forget gate gets rid of some values
  - the cell states is used additively (preventing values from getting smaller)
  - we only have positives values, since we are using sigmoid function
- use gradient clipping to prevent exploding
  - use a threshold on the gradients
  - if $||g|| > \text{threshold}$ $g = \frac{threshold * g}{||g||}$ where $||.||$ denotes the length
- weight regularization: L1 or L2

Vanishing and exploding gradients: ways to go

- residual (skip) connections
  - they allow gradient information to pass through the layers
  - creating "highways" of information
  - output of a previous layer is added to the output of a deeper layer
- change activation function
  - e.g ReLU only saturates in one direction (but now exploding is possible)
- use LSTM, GRU to prevent vanishing
  - the forget gate gets rid of some values
  - the cell states is used additively (preventing values from getting smaller)
  - we only have positives values, since we are using sigmoid function
- use gradient clipping to prevent exploding
  - use a threshold on the gradients
  - if $||g|| > \text{threshold}$ $g = \frac{threshold * g}{||g||}$ where $||.||$ denotes the length
- weight regularization: L1 or L2

Vanishing and exploding gradients: ways to go

- residual (skip) connections
  - they allow gradient information to pass through the layers
  - creating "highways" of information
  - output of a previous layer is added to the output of a deeper layer
- change activation function
  - e.g ReLU only saturates in one direction (but now exploding is possible)
- use LSTM, GRU to prevent vanishing
  - the forget gate gets rid of some values
  - the cell states is used additively (preventing values from getting smaller)
  - we only have positives values, since we are using sigmoid function
- use gradient clipping to prevent exploding
  - use a threshold on the gradients
  - if $||g|| > \text{threshold}$ $g = \frac{threshold * g}{||g||}$ where $||.||$ denotes the length
- weight regularization: L1 or L2

Vanishing and exploding gradients: ways to go

- residual (skip) connections
  - they allow gradient information to pass through the layers
  - creating "highways" of information
  - output of a previous layer is added to the output of a deeper layer
- change activation function
  - e.g ReLU only saturates in one direction (but now exploding is possible)
- use LSTM, GRU to prevent vanishing
  - the forget gate gets rid of some values
  - the cell states is used additively (preventing values from getting smaller)
  - we only have positives values, since we are using sigmoid function
- use gradient clipping to prevent exploding
  - use a threshold on the gradients
  - if $||g|| > \text{threshold}$ $g = \frac{threshold * g}{||g||}$ where $||.||$ denotes the length
- weight regularization: L1 or L2

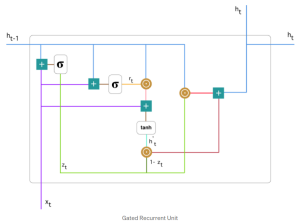
Vanishing and exploding gradients: ways to go

- residual (skip) connections
  - they allow gradient information to pass through the layers
  - creating "highways" of information
  - output of a previous layer is added to the output of a deeper layer
- change activation function
  - e.g ReLU only saturates in one direction (but now exploding is possible)
- use LSTM, GRU to prevent vanishing
  - the forget gate gets rid of some values
  - the cell states is used additively (preventing values from getting smaller)
  - we only have positives values, since we are using sigmoid function
- use gradient clipping to prevent exploding
  - use a threshold on the gradients
  - if $||g|| > \text{threshold}$ $g = \frac{threshold * g}{||g||}$ where $||.||$ denotes the length
- weight regularization: L1 or L2

Vanishing and exploding gradients: ways to go

- residual (skip) connections
  - they allow gradient information to pass through the layers
  - creating "highways" of information
  - output of a previous layer is added to the output of a deeper layer
- change activation function
  - e.g ReLU only saturates in one direction (but now exploding is possible)
- use LSTM, GRU to prevent vanishing
  - the forget gate gets rid of some values
  - the cell states is used additively (preventing values from getting smaller)
  - we only have positives values, since we are using sigmoid function
- use gradient clipping to prevent exploding
  - use a threshold on the gradients
  - if $||g|| > \text{threshold}$ $g = \frac{threshold * g}{||g||}$ where $||.||$ denotes the length
- weight regularization: L1 or L2

Gated Recurrent Units (GRU)

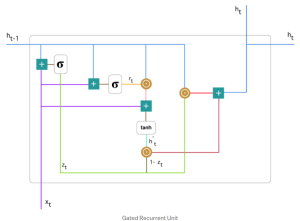


Gated Recurrent Unit

- are RNNs
- solve the vanishing gradient problem
- are a variation of LSTM
- perform quite good
- they use update gates (what to keep) and reset gates (what to forget)

see: https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be

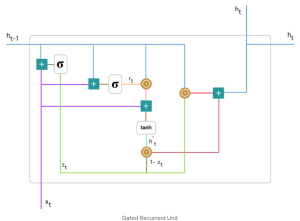
Gated Recurrent Units (GRU)



- are RNNs
- solve the vanishing gradient problem
- are a variation of LSTM
- perform quite good
- they use update gates (what to keep) and reset gates (what to forget)

see: https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be

Gated Recurrent Units (GRU)

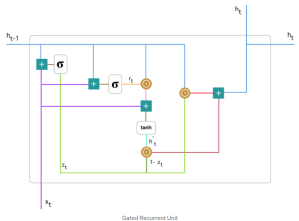


Gated Recurrent Unit

- are RNNs
- solve the vanishing gradient problem
- are a variation of LSTM
- perform quite good
- they use update gates (what to keep) and reset gates (what to forget)

see: https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be

Gated Recurrent Units (GRU)

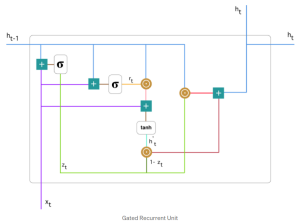


Gated Recurrent Unit

- are RNNs
- solve the vanishing gradient problem
- are a variation of LSTM
- perform quite good
- they use update gates (what to keep) and reset gates (what to forget)

see: https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be

Gated Recurrent Units (GRU)

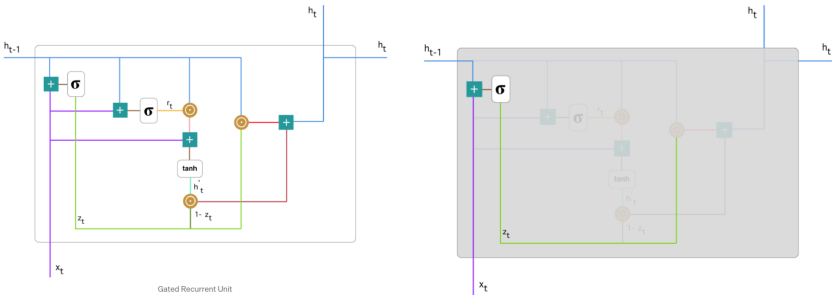


Gated Recurrent Unit

- are RNNs
- solve the vanishing gradient problem
- are a variation of LSTM
- perform quite good
- they use update gates (what to keep) and reset gates (what to forget)

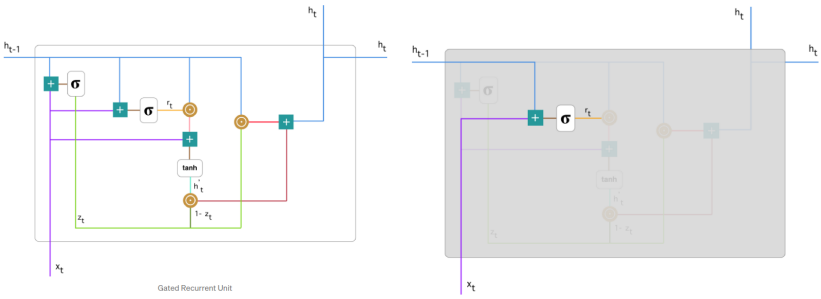
see: https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be

Gated Recurrent Units (GRU): Math. behind it



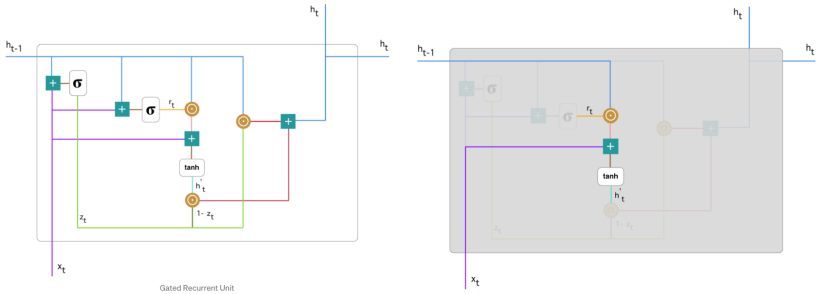
- update gate: $z_t = \sigma(W^z x_t + U^z h_{t-1})$
- $\sigma =$ sigmoid function

Gated Recurrent Units (GRU): Math. behind it



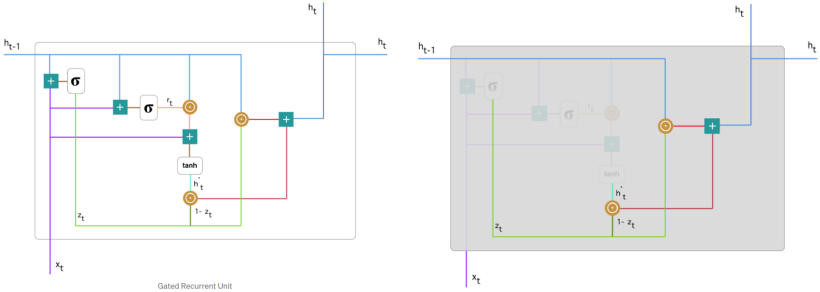
- update gate: $z_t = \sigma(W^z x_t + U^z h_{t-1})$
- reset gate: $r_t = \sigma(W^r x_t + U^r h_{t-1})$

Gated Recurrent Units (GRU): Math. behind it



- update gate: $z_t = \sigma(W^z x_t + U^z h_{t-1})$
- reset gate: $r_t = \sigma(W^r x_t + U^r h_{t-1})$
- current memory context: $h'_t = tanh(W x_t + r_t \odot U h_{t-1})$

Gated Recurrent Units (GRU): Math. behind it



- update gate: $z_t = \sigma(W^z x_t + U^z h_{t-1})$
- reset gate: $r_t = \sigma(W^r x_t + U^r h_{t-1})$
- current memory context: $h'_t = tanh(W x_t + r_t \odot U h_{t-1})$
- output: $h_t = z_t \odot h_{t-1} + (1 - z_t) \odot h'_t$

Overfitting the data

- if the model learns on the basis of idiosyncracies of the data
- if the model adopts to the noice in the data
- if the model takes the training data to seriously

Prevent overfitting by

- use a larger training set
- use a smaller network
- early stopping
- data normalization (might help)
- weight decay (regularization): penalize large weights
- model averaging
- dropout

Prevent overfitting

early stopping

- use (also) a validation set to determine the loss
- stop learning if it reaches a minimum (and afterwards starts to increase again)

Data normalization

i.e. scale the input features of a neural network, so that all features are scaled similarly

Dropout (only in training mode)

- goal: prevent nodes from creating to strong connections/dependencies among each other
- we randomly drop (zero) out portions of neurons from each training iteration

Regularization

high weights indicate overfitting

- here means: add the (scaled) length of the weight vector to the loss function
- the loss gets higher the larger the weights
- since we are minimizing, weights are kept small(er)

- L1 regularization: $\sum k|w_k|$
- mathematically, this encourages weights to be exactly 0

- L2 regularization: $\sum k\ w_k^2$
- mathematically, the weight is pushed towards 0

- combination of L1 and L2 regularization: add a term $\sum k|w_k| + \sum kw_k^2$ to the loss function

In PyTorch, a parameter weight_decay can be used e.g. in SGD and other optimizers