TECHNISCHE UNIVERSITÄT DRESDEN

SEMANTIC COMPUTING

Lecture 9: Deep Learning: Recurrent Neural Networks (RNNs)

Dagmar Gromann

International Center For Computational Logic

TU Dresden, 21 December 2018

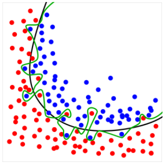
Overview

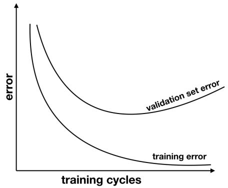
- Handling Overfitting
- Recurrent Neural Networks

Handling Overfitting

How to know your model overfits?

- Separate data into training, validation, and test set
- Train on training set only and evaluate on validation set
- Fine-tune hyperparameters and train again
- DO NOT touch the test set until the model is trained with a good performance on the training and test set
- Overfitting indicator: high performance on training set, low performance on validation set





Reduce Overfitting

Regularization

Any modification to a learning algorithm intended to reduce its generalization error (NOT the training error).

Regluarization methods:

- norm constraint
- dropout
- batch normalization
- data augmentation
- early stopping
- ...

Definition source: Goodfellow et al. (2016) Deep Learning, MIT Press. `http://www.deeplearningbook.org/`

Norm Constraint

- method for penalizing model complexity by adding a constraint to the cost function that balances against complexity
- minimize(Loss(Data | Model) + complexity(Model))
- L_2 regularization (ridge regularization):
  - penalize very big weights
  - add a constraint to the loss function
  - $L(w, \boldsymbol{x}) + \lambda ||w||_2^2$
    - w are weights and $\boldsymbol{x}$ are input data
    - where $\frac{2}{2}$ represents the sum of squares of all feature weights
    - L is the loss function
    - λ is the coefficient that regulates the tradeoff between correct prediction and model simplification

Dropout

Dropout

Powerful method to avoid overfitting by randomly ignoring subsets of neurons in a given layer during training, which means those neurons do not participate in producing a prediction for the data.

Why does that make sense?

A single neuron cannot fully rely on any one feature, so it is forced to spread its weights more, which has the effect of shrinking the squared norm of the weights.

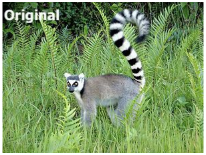
The higher the value in dropout, the more neurons you keep per layer (1.0 - no dropout at all).

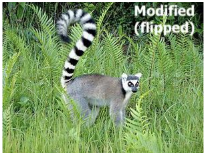
Data Augmentation

Data Augmentation

Add new training examples that are similar to the existing data but reasonably modified so that more variants can be learned.

Example of Modification





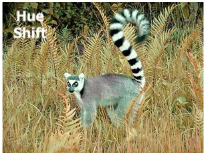


Image source: `https://en.wikipedia.org/wiki/Wildlife_of_Madagascar#/media/File:Maki.jpg`

Recurrent Neural Networks

Why “recurrent”?

- specialized for processing a sequence of inputs (of variable length)
- parameters are shared through different parts of the model across different time steps
- this parameter sharing allows information to persist across layers
- networks with loops, that is, repetitive (“recurring”) structures

Unfolding Recurrent Neural Networks

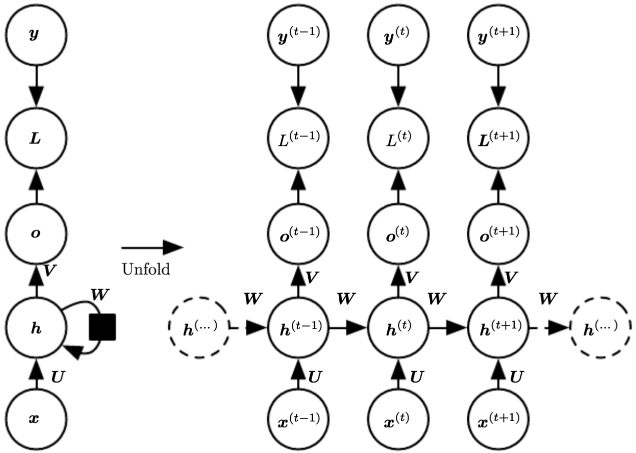


Image source: Goodfellow et al. (2016) Deep Learning, MIT Press. http://www.deeplearningbook.org/

Forward Propagation in this RNN

Assumptions:

- activation function on hidden layer is tanh
- output layer uses softmax
- output = discrete (e.g. words or characters)
- t represents the time steps: $t = 1, ..., \tau$
- U, W, V are weight matrices
- b and c are bias vectors

$$a^{(t)} = Wh^{(t-1)} + Ux^{(t)} + b$$

$$h^{(t)} = tanh(a^{(t)})$$

$$o^{(t)} = Vh^{(t)} + c$$

$$\hat{y}^{(t)} = softmax(o^{(t)})$$

$$L(\hat{y}, y) = -\sum_{t=1}^{\tau} y^{(t)} log \hat{y}^{(t)}$$

Networks with Output Recurrence

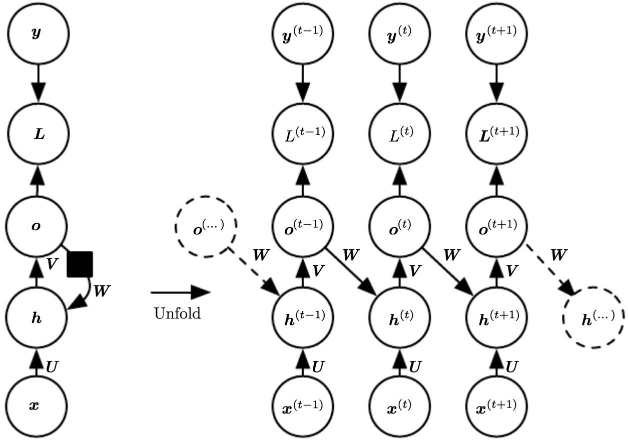


Image source: Goodfellow et al. (2016) Deep Learning, MIT Press. `http://www.deeplearningbook.org/`

Flexibility of Recurrent Neural Networks (RNNs)

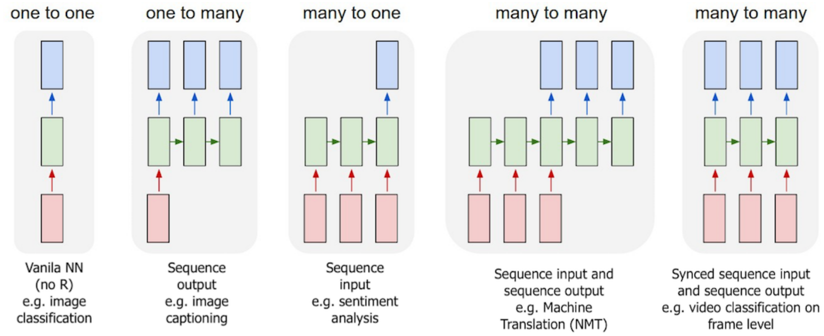

one to one
one to many
many to one
many to many
many to many
Vanila NN (no R) e.g. image classification
Sequence output e.g. image captioning
Sequence input e.g. sentiment analysis
Sequence input and sequence output e.g. Machine Translation (NMT)
Synced sequence input and sequence output e.g. video classification on frame level

Image adapted from Fei-Fei Li, Andrej Karpathy & Justin Johnson

Character-Level RNN Example

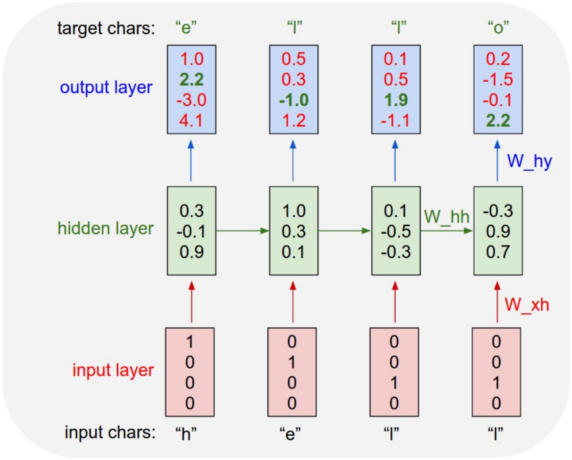


Image adapted from Fei-Fei Li, Andrej Karpathy & Justin Johnson

Backpropagation Through Time (BPTT)

Variant of backpropagation that works on a sequence in time:

- For each training epoch: start by training on shorter sequences and then progressively train on longer sequences
- For each length of sequence k: unfold the network (see previous slide) so that it resembles a normal feedforward network with k hidden layers
- Proceed with the standard backpropagation algorithm
- Issue: high cost of a single parameter update
- **Truncated BPTT**: process the sequence one timestep at a time, and every t_1 timesteps it runs BPTT for t_2 timesteps, which makes a parameter update cheaper if t_2 is small

Problems of RNNs

- Powerful but difficult to train
  - in backpropagation the gradient is multiplied a large number of times by the same weight matrix associated with the recurrent connections
  - if weights are small (< 1) => vanishing gradient problem
  - if weights are big (> 1) => exploding gradient problem
- Solution:
  - exploding gradient: clipping the norm of the exploded gradient when it is too large
  - vanishing gradient: relax non-linear dependency to linear dependency (LSTM, GRU, etc.)

Review of Lecture 9

- On which basis do cost function and activation function interact?
- What is backpropagation?
- What is the purpose of regularization in a neural network? How can this be achieved?
- What is a Recurrent Neural Network?
- How does backpropagation work with an RNN?
- What is a typical application scenario of an RNN?
- How can the problem of exploding gradients be treated in this type of network?