

Deep Learning for Text

Ayoub Bagheri

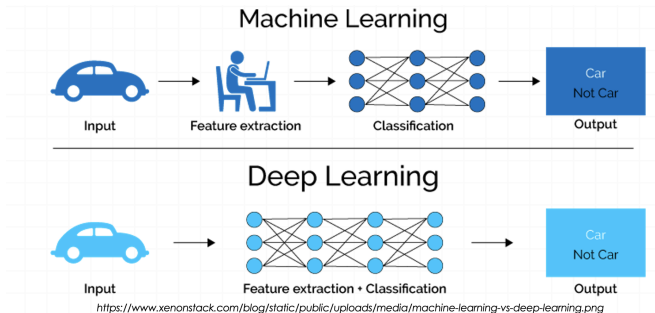
Lecture plan

1. Deep learning
2. Feed-forward neural networks
3. Recurrent neural networks

What is Deep Learning (DL)?

A machine learning subfield of learning representations of data. Exceptional effective at learning patterns.

Deep learning algorithms attempt to learn (multiple levels of) representation by using a hierarchy of multiple layers.

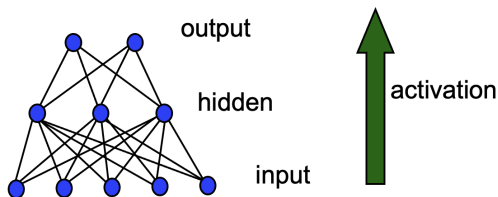


Deep learning vs neural networks

Deep learning architectures

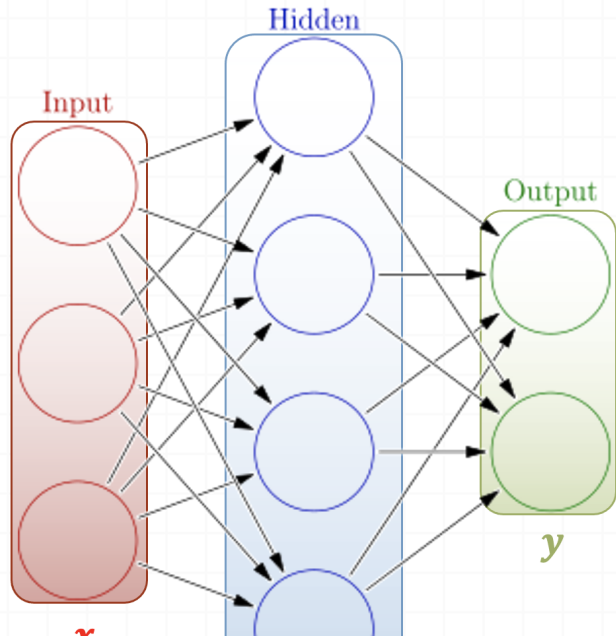
Feed-forward neural networks

- ▶ A typical multi-layer network consists of an input, hidden and output layer, each fully connected to the next, with activation feeding forward.

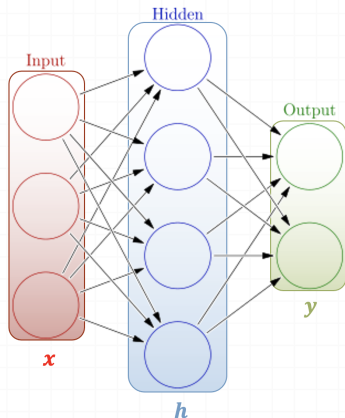


- ▶ The weights determine the function computed.

Feed-forward neural networks



Feed-forward neural networks



Weights

$$h = \sigma(W_1 x + b_1)$$
$$y = \sigma(W_2 h + b_2)$$

Activation functions

4 + 2 = 6 neurons (not counting inputs)
[3 x 4] + [4 x 2] = 20 weights
4 + 2 = 6 biases

26 learnable **parameters**

One forward pass

Text (input) representation

TFIDF

Word embeddings

....

0.2	-0.5	0.1
2.0	1.5	1.3
0.5	0.0	0.25
-0.3	2.0	0.0

W

0.1
0.2
0.3

x_i

+

1.0
3.0
0.025
0.0

b

=

0.95
3.89
0.15
0.37

$\sigma(x_i; W, b)$

very positive

positive

negative

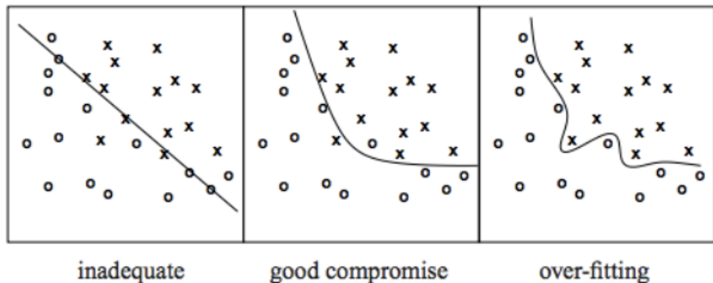
very negative

Hidden unit representations

- ▶ Trained hidden units can be seen as newly constructed features that make the target concept linearly separable in the transformed space.
- ▶ On many real domains, hidden units can be interpreted as representing meaningful features such as vowel detectors or edge detectors, etc..
- ▶ However, the hidden layer can also become a distributed representation of the input in which each individual unit is not easily interpretable as a meaningful feature.

Overfitting

Learned hypothesis may fit the training data very well, even outliers (noise) but fail to generalize to new examples (test data)



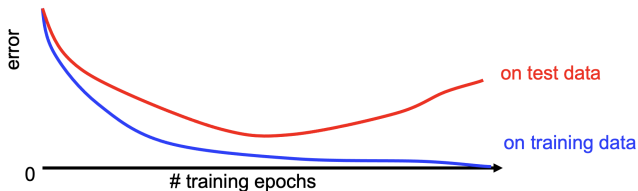
<http://wiki.bethanycrane.com/overfitting-of-data>

error



Overfitting prevention

- ▶ Running too many epochs can result in over-fitting.



- ▶ Keep a hold-out validation set and test accuracy on it after every epoch. Stop training when additional epochs actually increase validation error.
- ▶ To avoid losing training data for validation:
 - ▶ Use internal K-fold CV on the training set to compute the average number of epochs that maximizes generalization accuracy.
 - ▶ Train final network on complete training set for this many epochs.

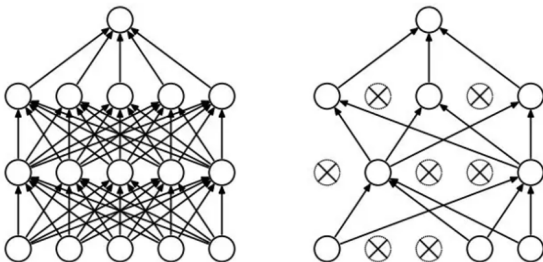
Regularization

Dropout

Randomly drop units (along with their connections) during training

Each unit retained with fixed probability p , independent of other units

Hyper-parameter p to be chosen (tuned)



Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting." *Journal of machine learning research* (2014)

Loss functions and output

Classification

Training examples

$\mathbb{R}^n \times \{\text{class}_1, \dots, \text{class}_n\}$
(one-hot encoding)

Output Layer

Soft-max
[map \mathbb{R}^n to a probability]

$$P(y = j \mid \mathbf{x}) = \frac{e^{\mathbf{x}^T \mathbf{w}_j}}{\sum_{k=1}^K e^{\mathbf{x}^T \mathbf{w}_k}}$$

Cost (loss) function

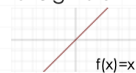
Cross-entropy

$$J(\theta) = -\frac{1}{n} \sum_{i=1}^n \sum_{k=1}^K [y_k^{(i)} \log \hat{y}_k^{(i)} + (1 - y_k^{(i)}) \log (1 - \hat{y}_k^{(i)})]$$

Regression

$\mathbb{R}^n \times \mathbb{R}^m$

Linear (Identity)
or Sigmoid



Mean Squared Error

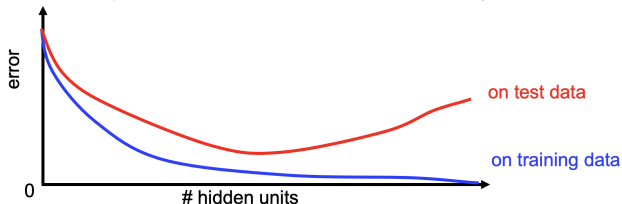
$$J(\theta) = \frac{1}{n} \sum_{i=1}^n (y^{(i)} - \hat{y}^{(i)})^2$$

Mean Absolute Error

$$J(\theta) = \frac{1}{n} \sum_{i=1}^n |y^{(i)} - \hat{y}^{(i)}|$$

Determining the best number of hidden units

- ▶ Too few hidden units prevents the network from adequately fitting the data.
- ▶ Too many hidden units can result in over-fitting.



- ▶ Use internal cross-validation to empirically determine an optimal number of hidden units.
- ▶ Hyperparameter tuning

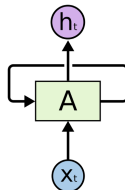
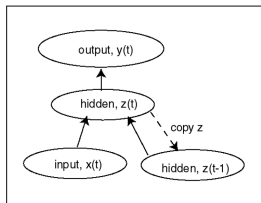
Recurrent Neural Networks

Recurrent Neural Network (RNN)

- ▶ Add feedback loops where some units' current outputs determine some future network inputs.
- ▶ RNNs can model dynamic finite-state machines, beyond the static combinatorial circuits modeled by feed-forward networks.

Simple Recurrent Network (SRN)

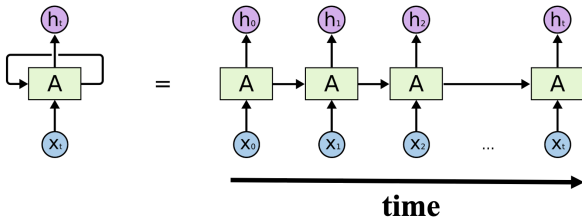
- ▶ Initially developed by Jeff Elman (“*Finding structure in time*,” 1990).
- ▶ Additional input to hidden layer is the state of the hidden layer in the previous time step.



<http://colah.github.io/posts/2015-08-Understanding-LSTMs/>

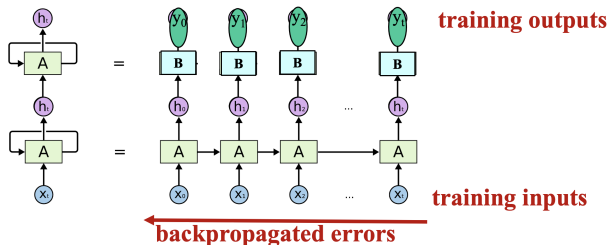
Unrolled RNN

- Behavior of RNN is perhaps best viewed by “unrolling” the network over time.



Training RNNs

- ▶ RNNs can be trained using “backpropagation through time.”
- ▶ Can viewed as applying normal backprop to the unrolled network.



LSTM

Vanishing gradient problem

Suppose we had the following scenario:

Day 1: Lift Weights

Day 2: Swimming

Day 3: At this point, our model must decide whether we should take a rest day or yoga. Unfortunately, it only has access to the previous day. In other words, it knows we swam yesterday but it doesn't know whether had taken a break the day before.

Therefore, it can end up predicting yoga.

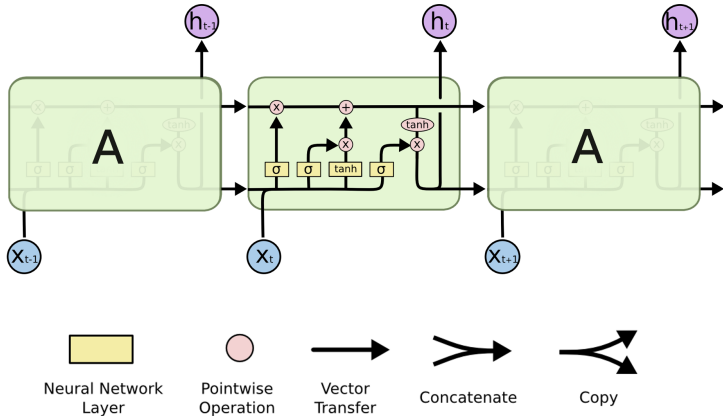
- ▶ Backpropagated errors multiply at each layer, resulting in exponential decay (if derivative is small) or growth (if derivative is large).
- ▶ Makes it very difficult train deep networks, or simple recurrent networks over many time steps.
- ▶ LSTMs were invented, to get around this problem.

<https://towardsdatascience.com/>

Long Short Term Memory

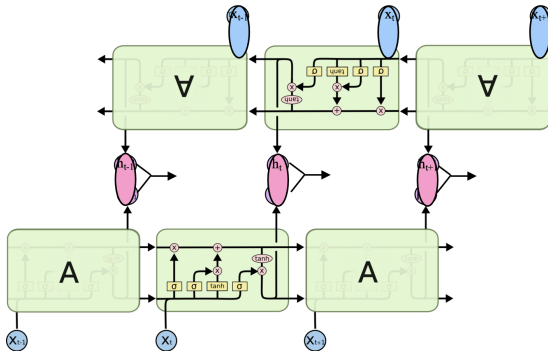
- ▶ LSTM networks, add additional gating units in each memory cell.
 - ▶ Forget gate
 - ▶ Input gate
 - ▶ Output gate
- ▶ Prevents vanishing/exploding gradient problem and allows network to retain state information over longer periods of time.

LSTM network architecture | <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>



Bi-directional LSTM (Bi-LSTM)

- ▶ Separate LSTMs process sequence forward and backward and hidden layers at each time step are concatenated to form the cell output.



Advanced models

- ▶ For many applications, it helps to add “attention” to RNNs.
- ▶ Allows network to learn to attend to different parts of the input at different time steps, shifting its attention to focus on different aspects during its processing.
- ▶ Used in image captioning to focus on different parts of an image when generating different parts of the output sentence.
- ▶ In MT, allows focusing attention on different parts of the source sentence when generating different parts of the translation.

Summary

Summary

- ▶ Deep learning
- ▶ Feed-forward neural networks
- ▶ Recurrent neural networks

Practical 8