Deep Learning for Text 1

Applied Text Mining

Ayoub Bagheri

Table of Contents

Lecture plan

1. Language modeling
2. Feed-forward neural networks
3. Recurrent neural networks

What is a Language Model?

A Language Model is a statistical tool that uses algorithms to predict the next word or sequence of words in a sentence. These models are fundamental in natural language processing (NLP) and are used to understand and generate human language.

Key Points:

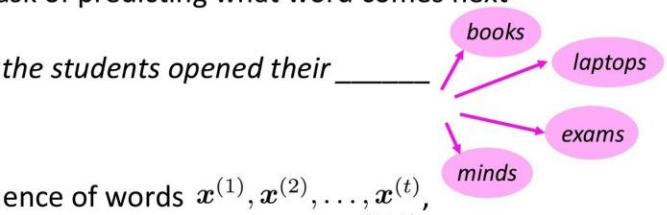
- Predictive Power: Language models can predict the next word based on the context of previous words.
- Applications: Used in machine translation, speech recognition, text generation, and more.
- Training Data: Trained on large datasets containing text, learning the structure and nuances of the language.

Types of Models:

- n-gram Models: Predict words based on the previous ‘n’ words.
- Neural Network Models: Use deep learning to understand and generate text, such as GPT (Generative Pre-trained Transformer).

Language Modeling

- **Language Modeling** is the task of predicting what word comes next



- More formally: given a sequence of words $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(t)}$, compute the probability distribution of the next word $\boldsymbol{x}^{(t+1)}$:

$$P(\boldsymbol{x}^{(t+1)} | \ \boldsymbol{x}^{(t)}, \ldots, \boldsymbol{x}^{(1)})$$

  where $\boldsymbol{x}^{(t+1)}$ can be any word in the vocabulary $V = \{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_{|V|}\}$

- A system that does this is called a **Language Model**

source: http://web.stanford.edu/class/cs224n/

Language Modeling

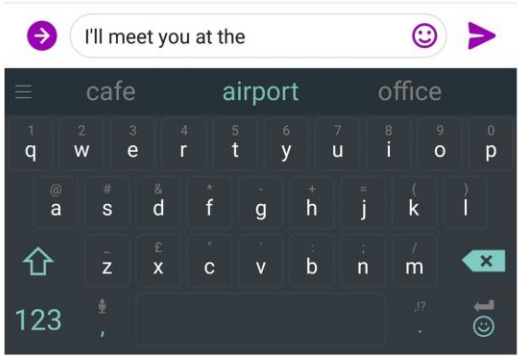
- You can also think of a Language Model as a system that assigns a probability to a piece of text

- For example, if we have some text $\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(T)}$, then the probability of this text (according to the Language Model) is:

$$P(\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(T)}) = P(\boldsymbol{x}^{(1)}) \times P(\boldsymbol{x}^{(2)} | \ \boldsymbol{x}^{(1)}) \times \cdots \times P(\boldsymbol{x}^{(T)} | \ \boldsymbol{x}^{(T-1)}, \ldots, \boldsymbol{x}^{(1)})$$
$$= \prod_{t=1}^{T} \underbrace{P(\boldsymbol{x}^{(t)} | \ \boldsymbol{x}^{(t-1)}, \ldots, \boldsymbol{x}^{(1)})}$$

This is what our LM provides

You use Language Models every day!



You use Language Models every day!

Google

what is the

what is the **weather**
what is the **meaning of life**
what is the **dark web**
what is the **xfl**
what is the **doomsday clock**
what is the **weather today**
what is the **keto diet**
what is the **american dream**
what is the **speed of light**
what is the **bill of rights**

Google Search I'm Feeling Lucky

n-gram Language Models

the students opened their _____

- **Question**: How to learn a Language Model?
- **Answer** (pre- Deep Learning): learn an *n*-gram Language Model!

- **Definition:** An *n*-gram is a chunk of *n* consecutive words.
  - unigrams: "the", "students", "opened", "their"
  - bigrams: "the students", "students opened", "opened their"
  - trigrams: "the students opened", "students opened their"
  - four-grams: "the students opened their"

- **Idea:** Collect statistics about how frequent different n-grams are and use these to predict next word.

n-gram Language Models

- First we make a Markov assumption: $x^{(t+1)}$ depends only on the preceding *n*-1 words

$$P(\boldsymbol{x}^{(t+1)}|\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(1)}) = P(\boldsymbol{x}^{(t+1)}|\overbrace{\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(t-n+2)}}^{n\text{-1 words}}) \quad \text{(assumption)}$$

prob of a n-gram

prob of a (n-1)-gram

$$= \frac{P(\boldsymbol{x}^{(t+1)},\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(t-n+2)})}{P(\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(t-n+2)})} \quad \text{(definition of conditional prob)}$$

- **Question:** How do we get these *n*-gram and (*n*-1)-gram probabilities?
- **Answer:** By counting them in some large corpus of text!

$$\approx \frac{\text{count}(\boldsymbol{x}^{(t+1)},\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(t-n+2)})}{\text{count}(\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(t-n+2)})} \quad \text{(statistical approximation)}$$

n-gram Language Models: Example

Suppose we are learning a 4-gram Language Model.

~~as the proctor started the clock, the~~ *students opened their* _____

discard

condition on this

$$P(\boldsymbol{w}|\text{students opened their}) = \frac{\text{count}(\text{students opened their } \boldsymbol{w})}{\text{count}(\text{students opened their})}$$

For example, suppose that in the corpus:

- "students opened their" occurred 1000 times
- "students opened their books" occurred 400 times
  - → P(books | students opened their) = 0.4
- "students opened their exams" occurred 100 times
  - → P(exams | students opened their) = 0.1

Should we have discarded the "proctor" context?

Generating text with a n-gram Language Model

You can also use a Language Model to generate text

today the price of gold per ton , while production of shoe lasts and shoe industry , the bank intervened just after it considered and rejected an imf demand to rebuild depleted european stocks , sept 30 end primary 76 cts a share .

Surprisingly grammatical!

…but **incoherent.** We need to consider more than three words at a time if we want to model language well.

How to build a *neural* language model?

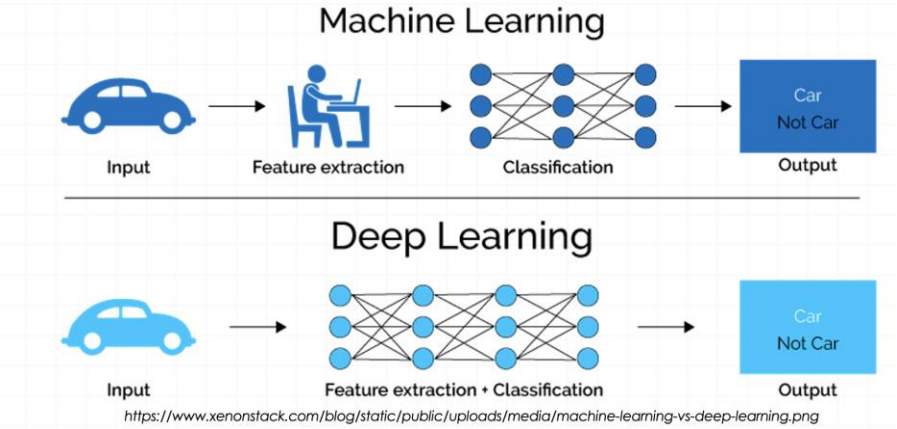
- Recall the Language Modeling task:
  - Input: sequence of words $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(t)}$
  - Output: prob. dist. of the next word $P(\boldsymbol{x}^{(t+1)} | \boldsymbol{x}^{(t)}, \ldots, \boldsymbol{x}^{(1)})$

Deep Learning

What is Deep Learning?

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

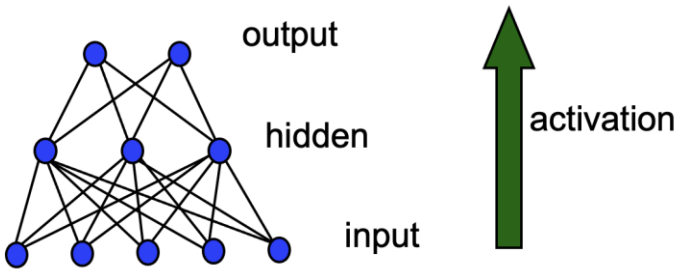
Deep learning algorithms attempt to learn (multiple levels of) representation by using a hierarchy of multiple layers.



https://www.xenonstack.com/blog/static/public/uploads/media/machine-learning-vs-deep-learning.png

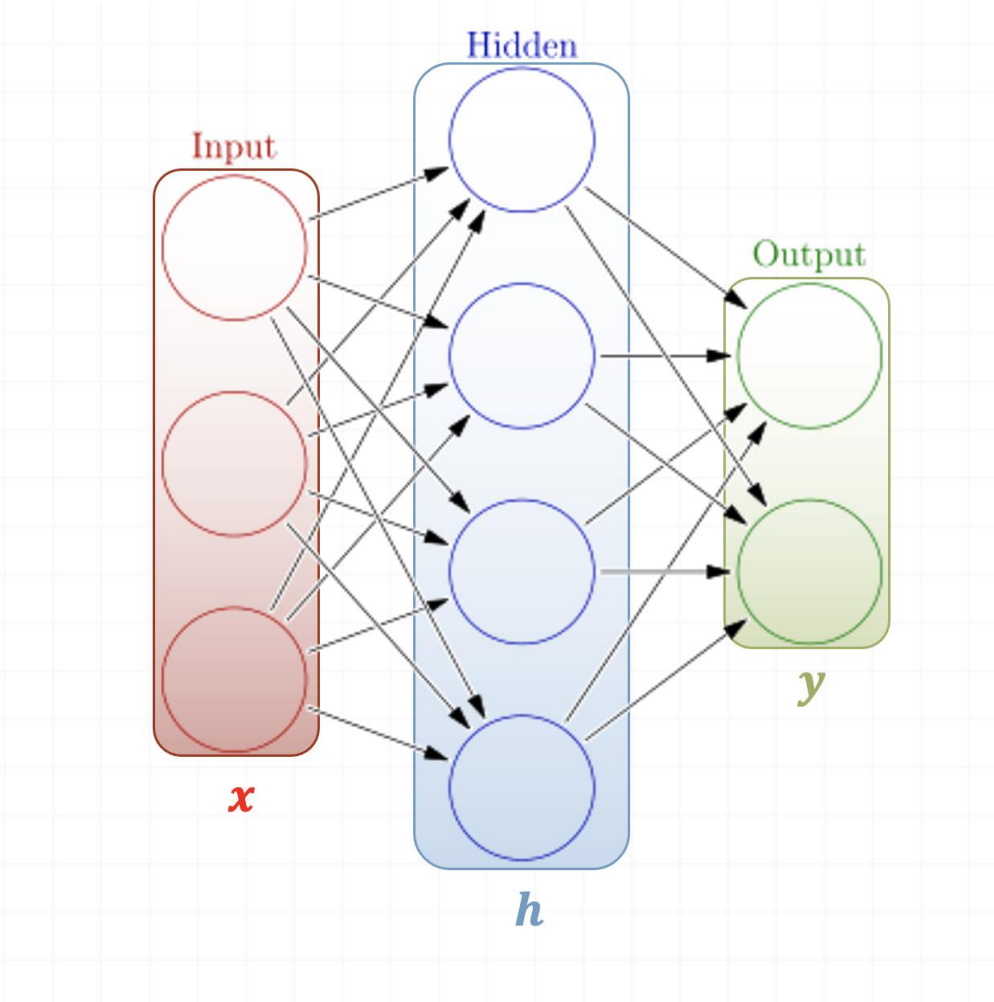
Feed-forward neural networks

- 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.

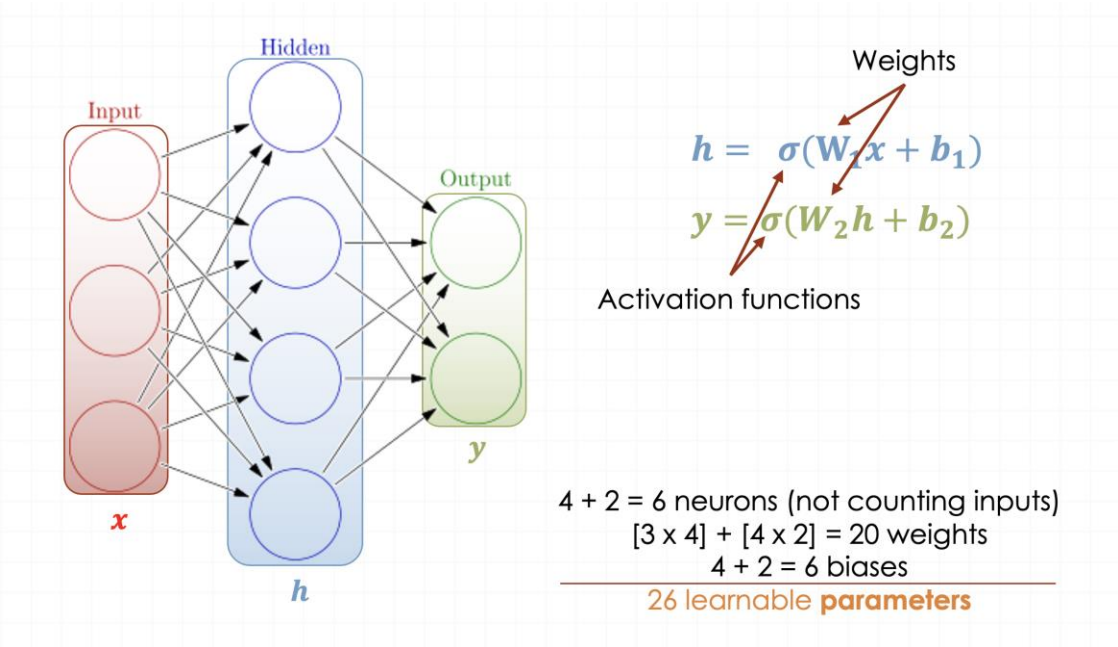
Feed-forward neural networks



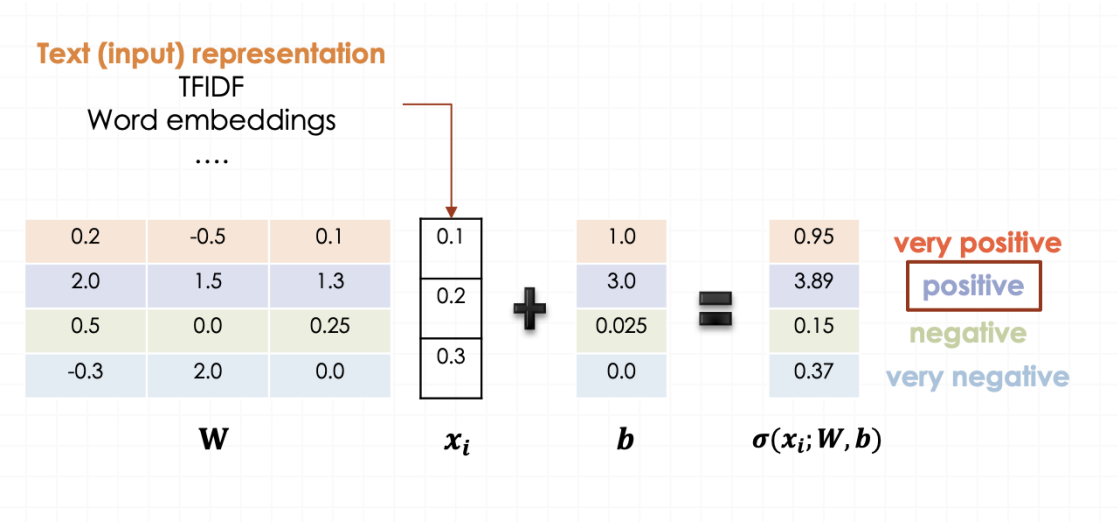
$$h = \sigma(W_1 x + b_1)$$

$$y = \sigma(W_2 h + b_2)$$

Feed-forward neural networks



One forward pass



Training

https://medium.com/@ramrajchandradevan/the-evolution-of-gradient-descend-optimization-algorithm-4106a6702d39

Optimize objective/cost function $J(\theta)$

Generate error signal that measures difference between predictions and target values

Use error signal to change the weights and get more accurate predictions

Subtracting a fraction of the gradient moves you towards the (local) minimum of the cost function

Updating weights

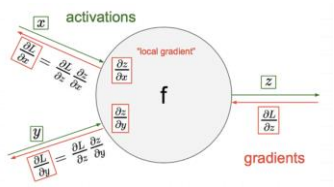
objective/cost function $J(\theta)$

Update each element of θ:

$$\theta_j^{new} = \theta_j^{old} - \alpha \frac{d}{\theta_j^{old}} J(\theta)$$

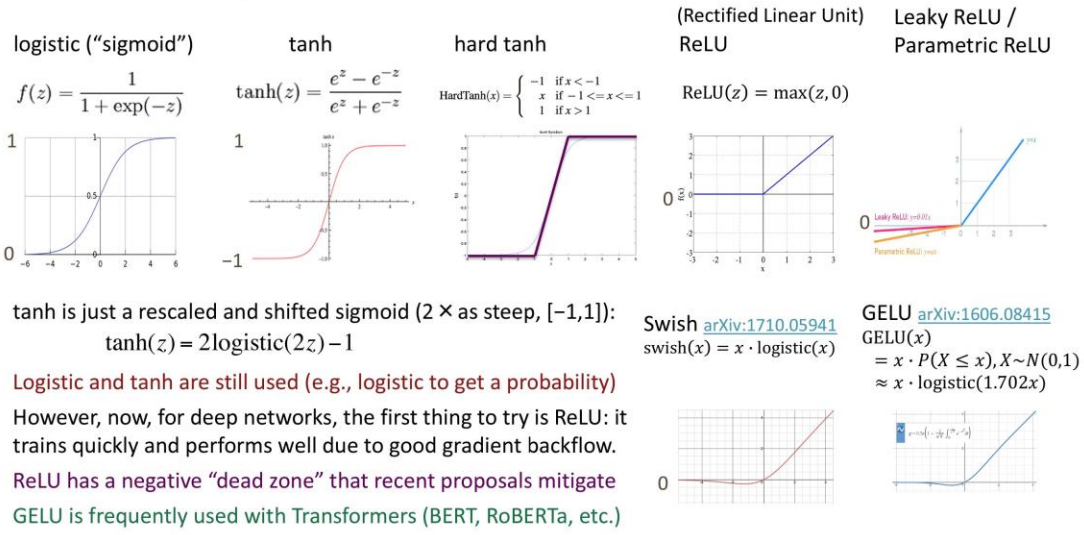
Matrix notation for all parameters (α: learning rate):

$$\theta_j^{new} = \theta_j^{old} - \alpha \nabla_\theta J(\theta)$$



Recursively apply chain rule though each node

Non-linearities, old and new



Notes on training

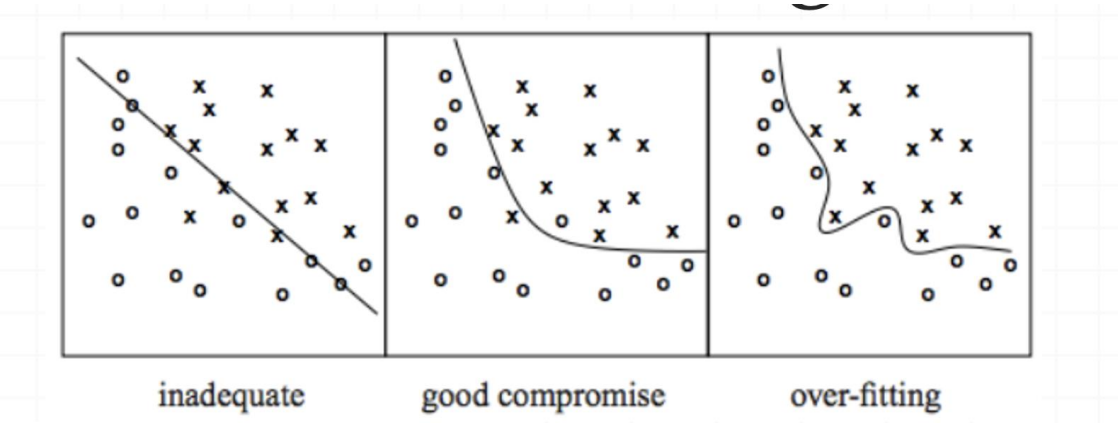
- Not guaranteed to converge to zero training error, may converge to local optima or oscillate indefinitely.
- However, in practice, does converge to low error for many large networks on real data.
- Many epochs (thousands) may be required, hours or days of training for large networks.
- **To avoid local-minima problems**, run several trials starting with different random weights (*random restarts*).
  - Take results of trial with lowest training set error.
  - Build a committee of results from multiple trials (possibly weighting votes by training set accuracy).

Hidden unit representations

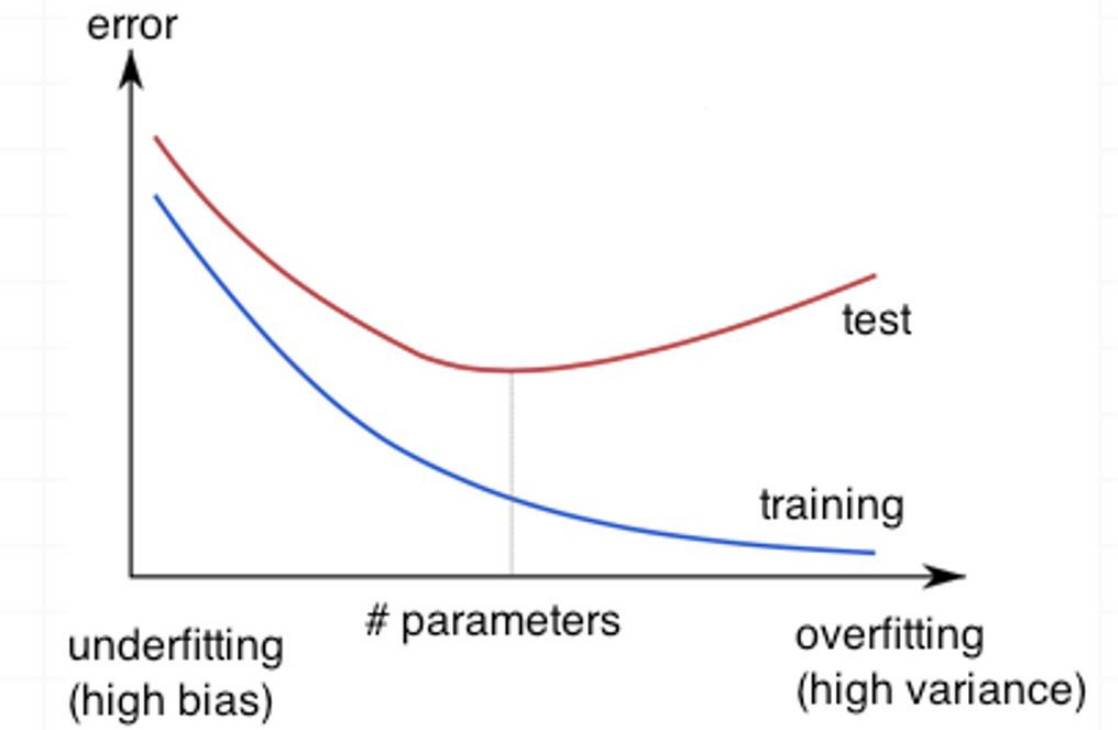
- Trained hidden units can be seen as **newly constructed features** that make the target concept linearly separable in the transformed space.
- On many real domains, hidden units can be interpreted as representing meaningful features such as vowel detectors or edge detectors, etc..
- However, the hidden 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

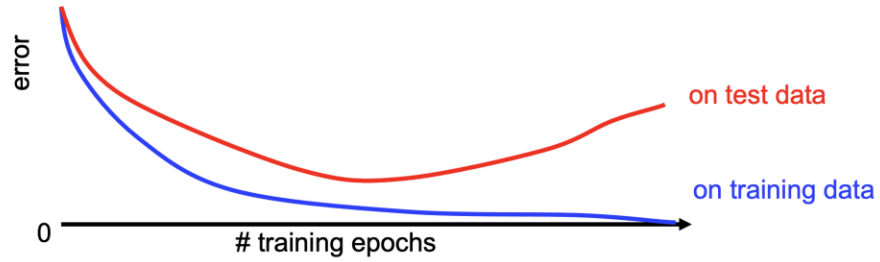


https://www.neuraldesigner.com/images/learning/selection_error.svg

How to avoid overfitting?

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.

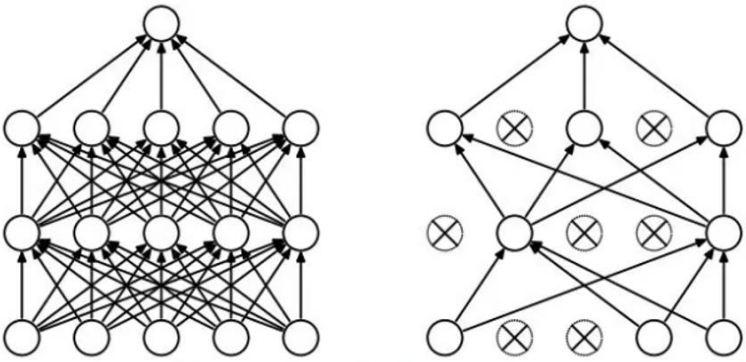
Regularization

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)

L2 = weight decay

Regularization term that penalizes big weights, added to the objective $J_{reg}(\theta) = J(\theta) + \lambda \sum_k \theta_k^2$

Weight decay value determines how dominant regularization is during gradient computation

Big weight decay coefficient &rarr big penalty for big weights

Early-stopping

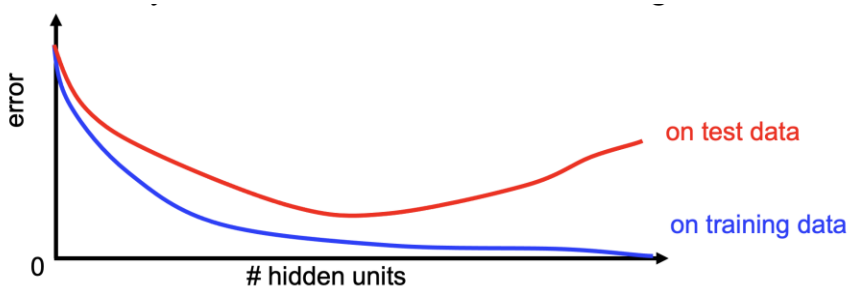
Use validation error to decide when to stop training

Stop when monitored quantity has not improved after n subsequent epochs

n is called patience

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

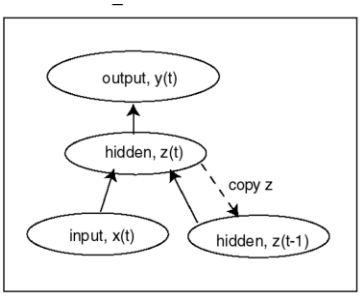
Recurrent Neural Networks

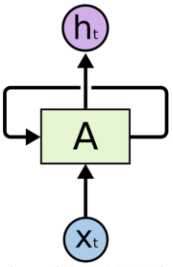
Recurrent Neural Network (RNN)

- Another architecture of NN
- RNN for LM
- 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.

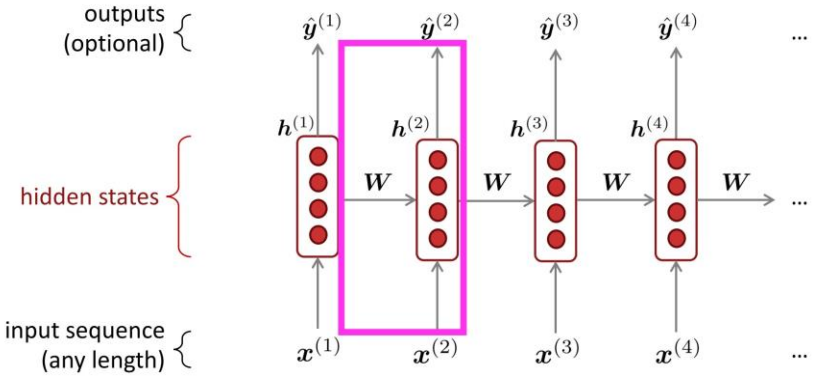




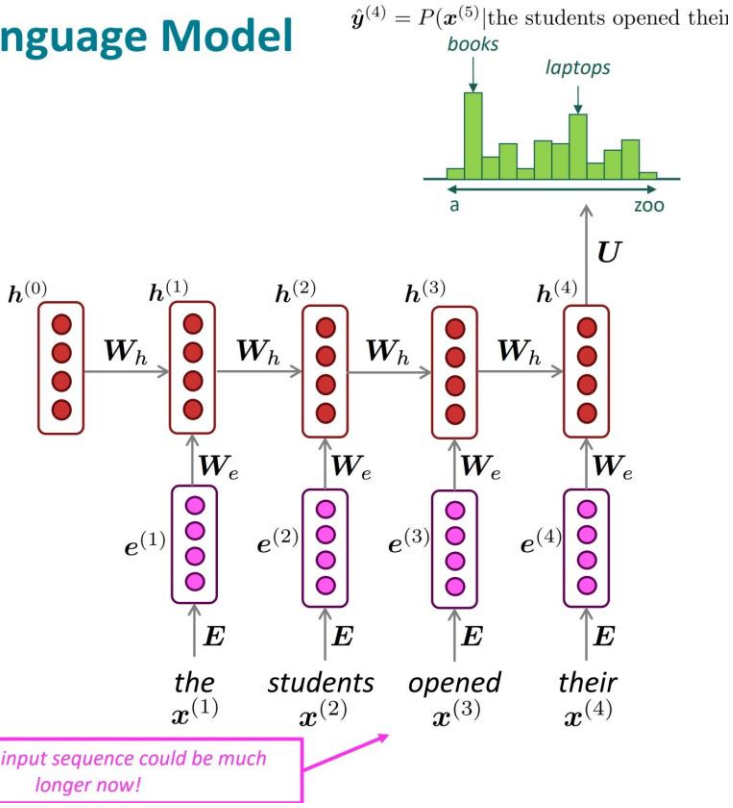
http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Unrolled RNN

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



A Simple RNN Language Model



output distribution

$\hat{y}^{(t)} = \text{softmax}\left(Uh^{(t)} + b_2\right) \in \mathbb{R}^{|V|}$

hidden states

$h^{(t)} = \sigma\left(W_h h^{(t-1)} + W_e e^{(t)} + b_1\right)$

$h^{(0)}$ is the initial hidden state

word embeddings

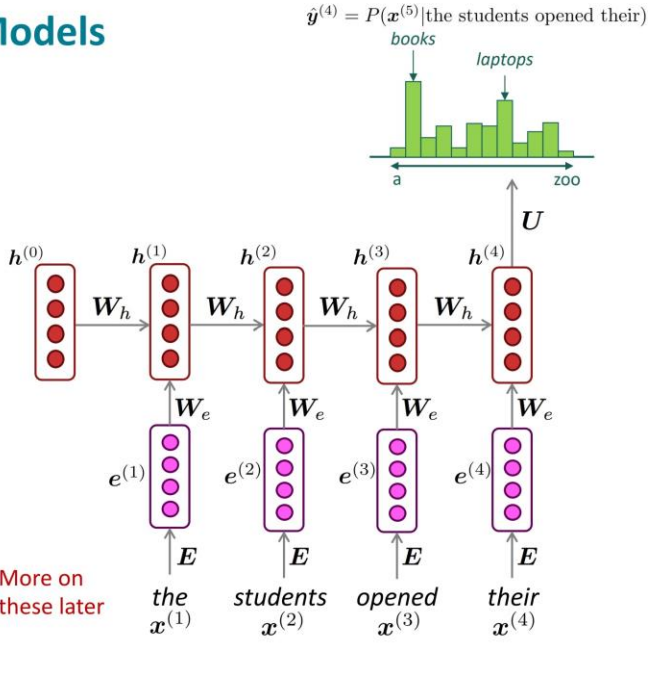
$e^{(t)} = Ex^{(t)}$

words / one-hot vectors

$x^{(t)} \in \mathbb{R}^{|V|}$

Note: *this input sequence could be much longer now!*

RNN Language Models



RNN **Advantages**:

- Can process any length input
- Computation for step *t* can (in theory) use information from many steps back
- Model size doesn't increase for longer input context
- Same weights applied on every timestep, so there is symmetry in how inputs are processed.

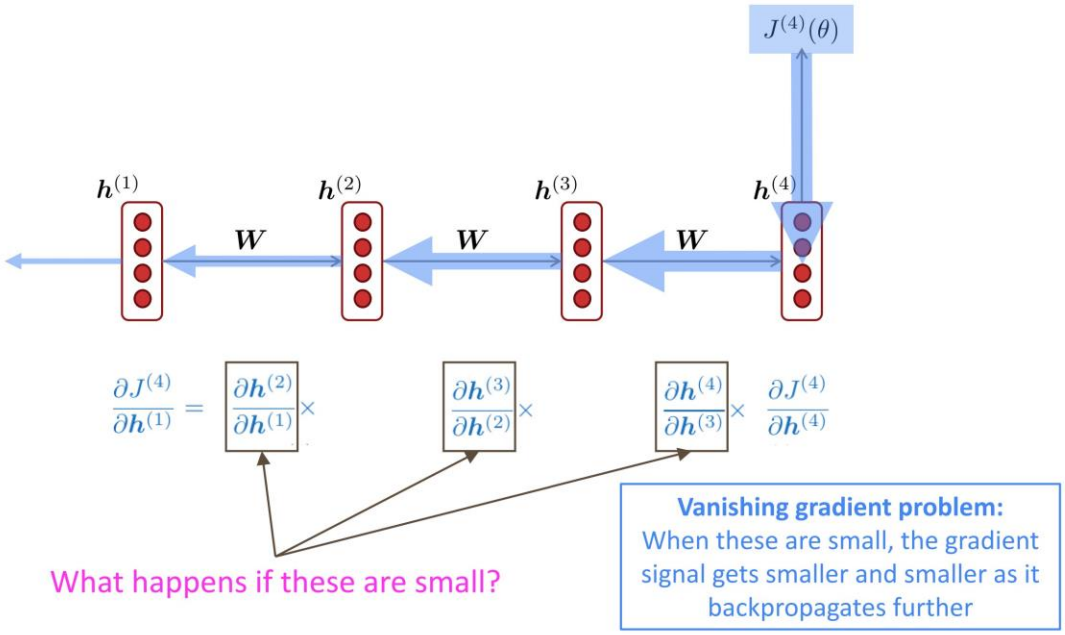
RNN **Disadvantages**:

- Recurrent computation is slow
- In practice, difficult to access information from many steps back

More on these later

LSTM

Vanishing gradient problem



Vanishing gradient problem

- **LM task:** *When she tried to print her tickets, she found that the printer was out of toner. She went to the stationery store to buy more toner. It was very overpriced. After installing the toner into the printer, she finally printed her* _____

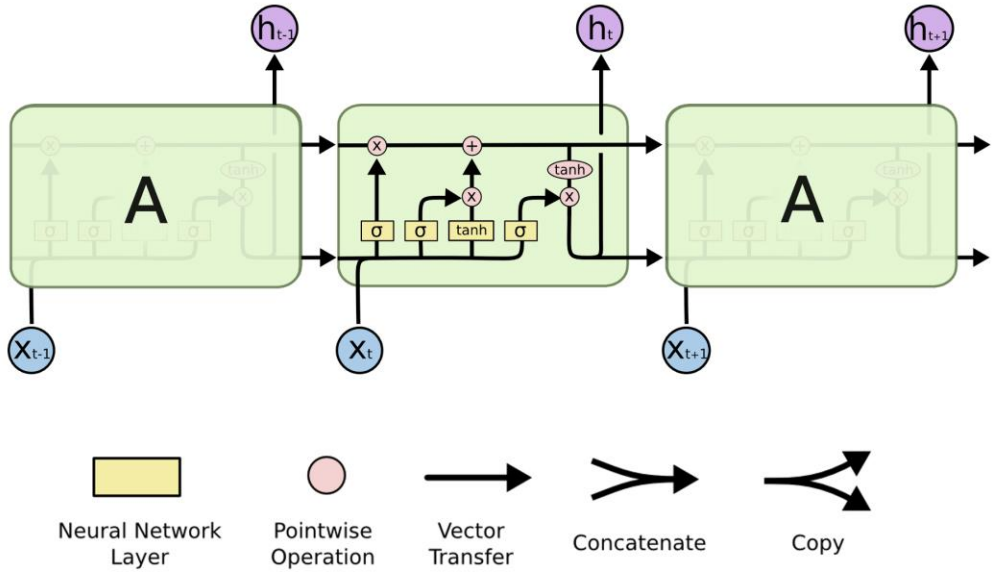
- To learn from this training example, the RNN-LM needs to model the dependency between *"tickets"* on the 7th step and the target word *"tickets"* at the end.

- But if the gradient is small, the model can't learn this dependency
  - So, the model is unable to predict similar long-distance dependencies at test time

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/



–>

–> –> –>

–>

–> –>

–> –> –>

–>

–>

–> –> –>

–>

–> –> –> –> –> –>

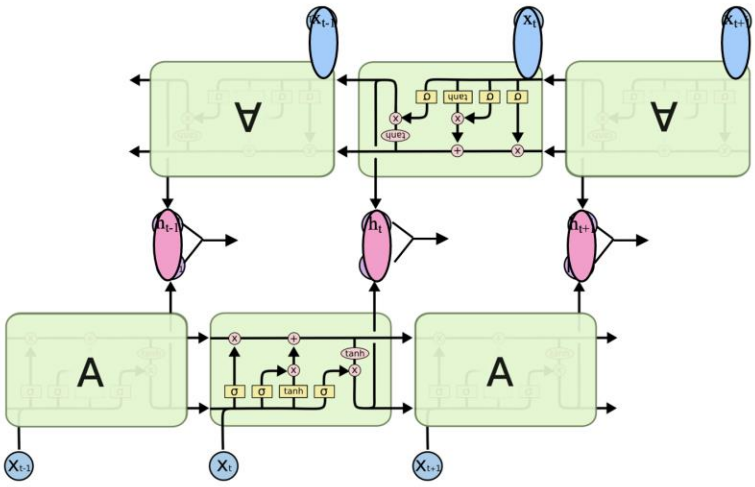
–>

–> –> –>

–> –> –> –> –> –> –> –>

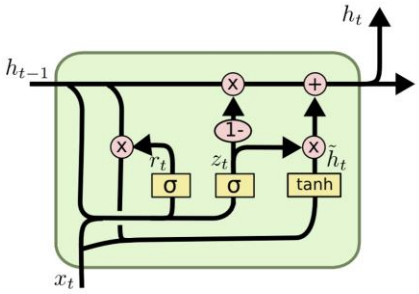
Bi-directional LSTM (Bi-LSTM)

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



Gated Recurrent Unit (GRU)

- Alternative RNN to LSTM that uses fewer gates (Cho, et al., 2014)
  - Combines forget and input gates into "update" gate.
  - Eliminates cell state vector



$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t])$$

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t])$$

$$\tilde{h}_t = tanh(W \cdot [r_t * h_{t-1}, x_t])$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{(h)}_t$$

Attention

- 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

- Language Model: A system that predicts the next word
- Deep learning can be applied for automatic feature engineering
- Recurrent Neural Network: A family of deep learning / neural networks that: • Take sequential input (Text) of any length; apply the same weights on each step • Can optionally produce output on each step
- Recurrent Neural Network ≠ Language Model
- RNNs can be used for many other things
- Language modeling can be done with different models, e.g., n-grams or transformers: GPT is an LM!

Practical 6