Applied Artificial Intelligence

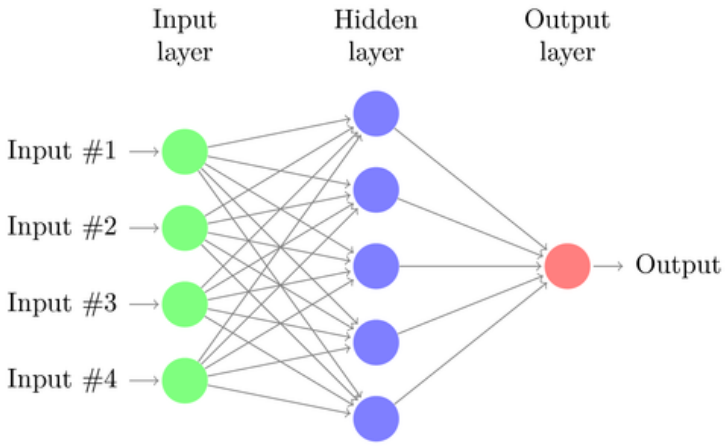
Session 20: DNN for Regression, Advanced Optimization, CNNs, and Motivation for RNNs

Fall 2018

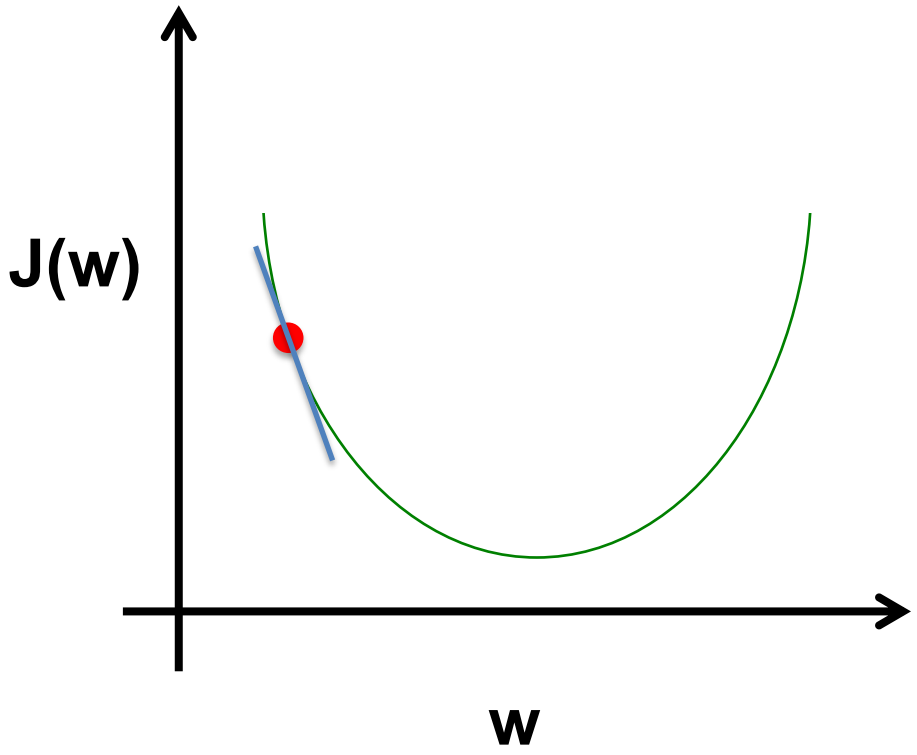
NC State University
Lecturer: Dr. Behnam Kia

Course Website: https://appliedai.wordpress.ncsu.edu/

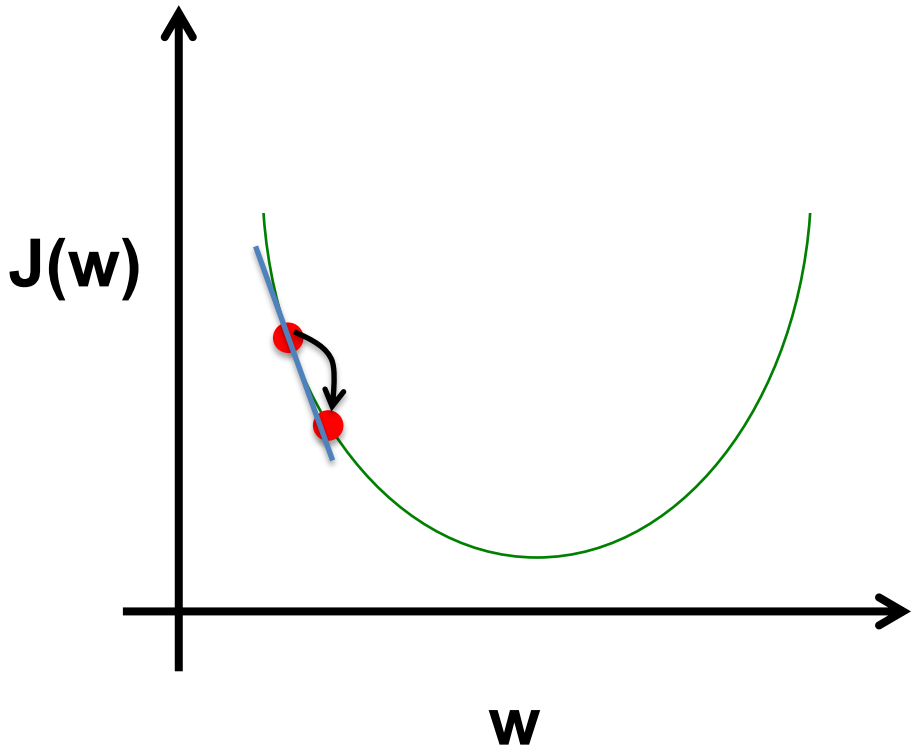
Feedforward networks for Regression

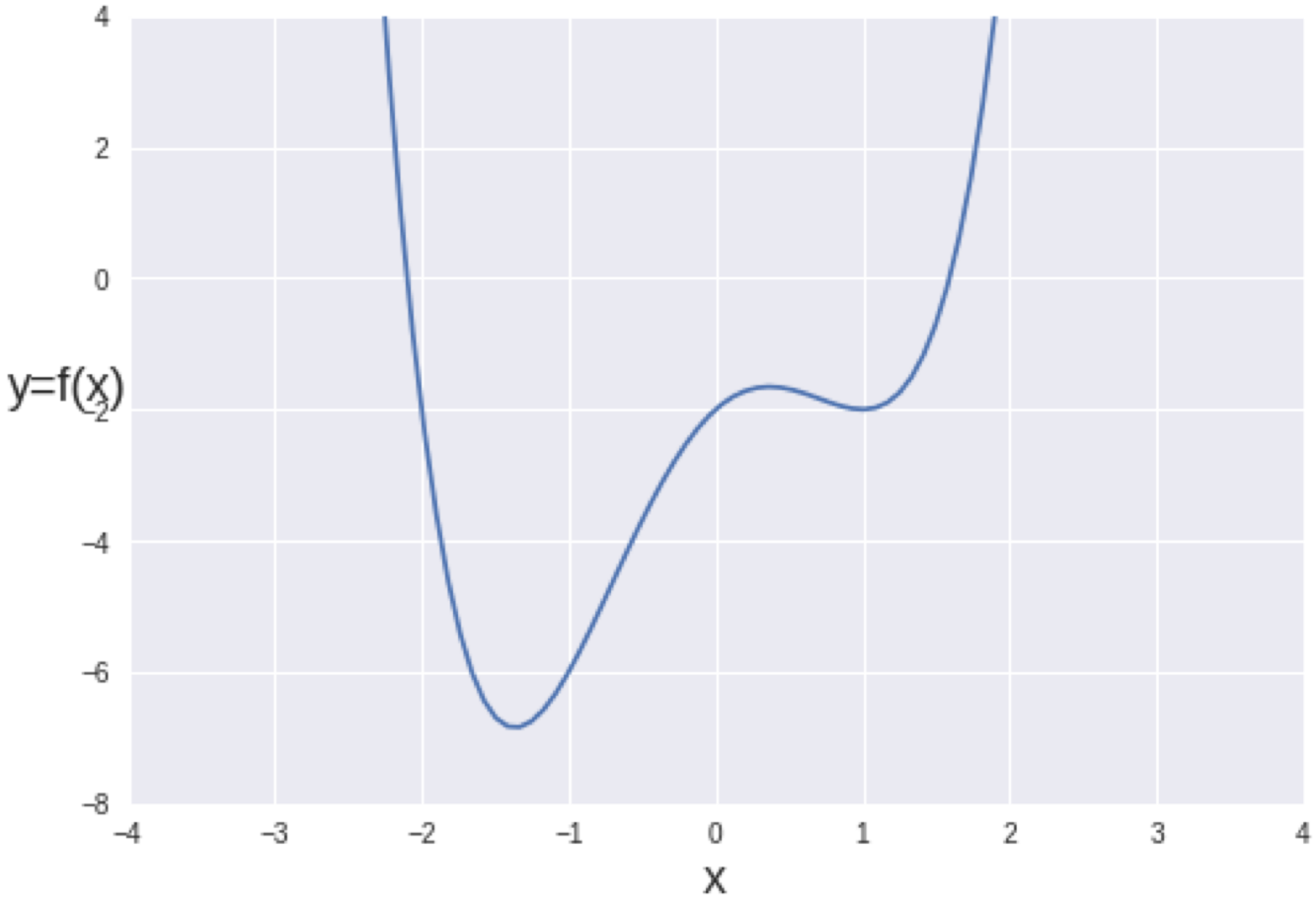


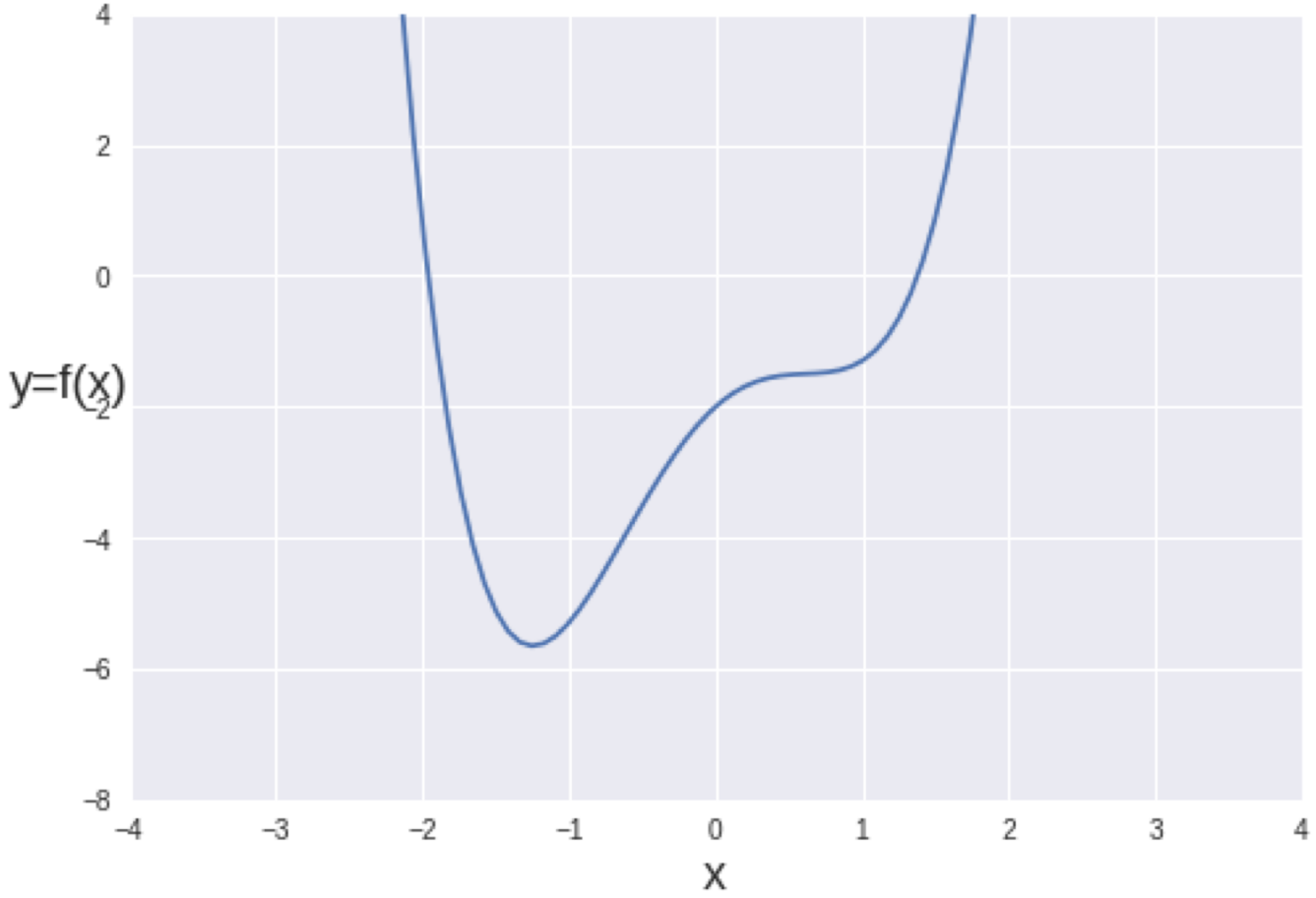
Gradient Descent



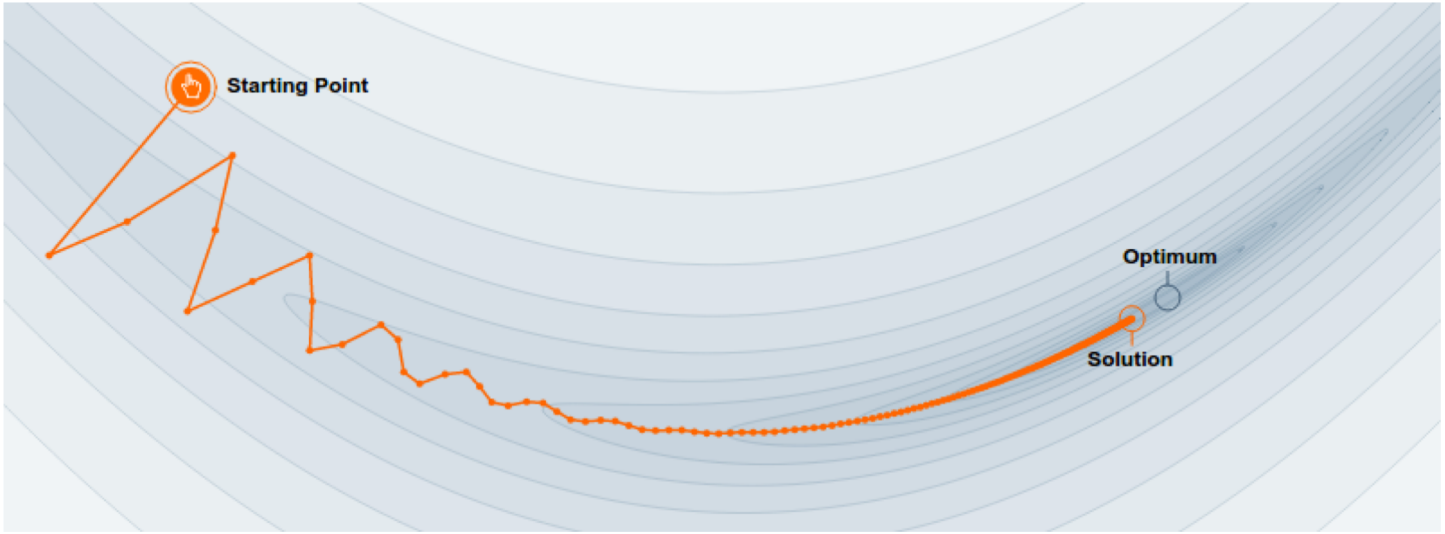
Gradient Descent

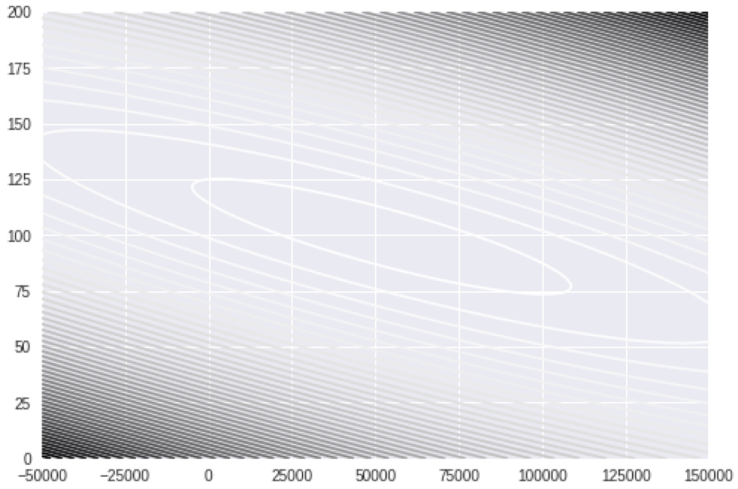


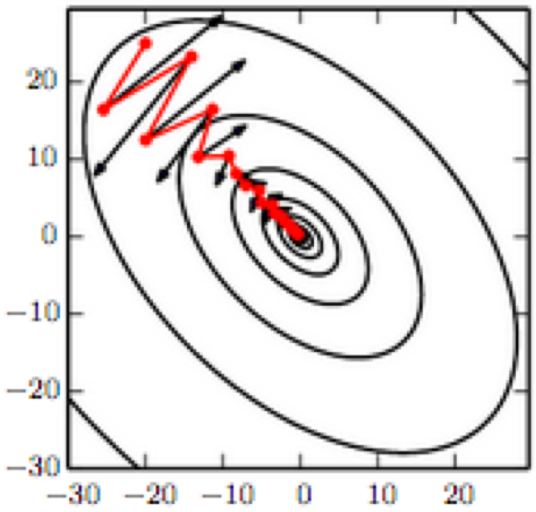

4
2
0
−2
−4
−6
−8
y=f(x)
−4
−3
−2
−1
0
1
2
3
4
X


y=f(x)
x
4
2
0
−2
−4
−6
−8
−4
−3
−2
−1
0
1
2
3
4

Momentum


Starting Point
Optimum
Solution


200
175
150
125
100
75
50
25
0
−50000
−25000
0
25000
50000
75000
100000
125000
150000


20
10
0
−10
−20
−30
−30
−20
−10
0
10
20

Momentum

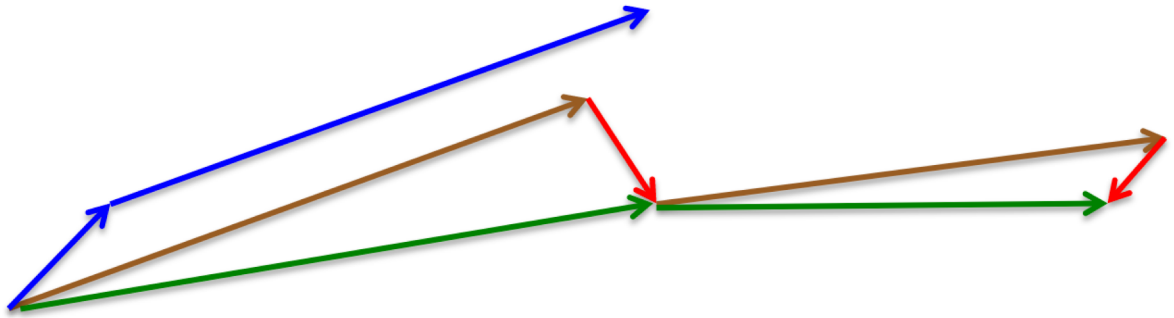
- Exponentially decaying moving average of the gradients
- $Numerical\ Example$:

$$v = -[0.9^1 \times g_{t-1} + 0.9^2 \times g_{t-2} + \cdots + 0.9^1 \times g_{t-10} + (1 - 0.9) \times g_t]$$

- g_i is gradient at step *i*
- Window size is 10 here, it could be infinity. With exponentially decaying weights it won't differ too much.
- $v_t = 0.9 \times v_{t-1} - (1 - 0.9) \times g_t$
- Instead of 0.9 we can take other numbers between 0 and 1.
- The closer to 1, the higher the momentum.

A picture of the Nesterov method

- First make a big jump in the direction of the previous accumulated gradient.
- Then measure the gradient where you end up and make a correction.



brown vector = jump, red vector = correction, green vector = accumulated gradient

blue vectors = standard momentum

From Geoffrey Hinton's lectures at UT

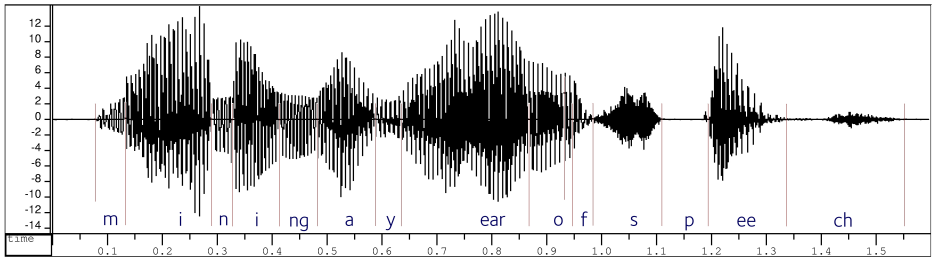
RMSProp

- **RMSProp**: Adapting step size separately for each parameter.
  - For each direction (parameter) divide the gradient by a running average of its recent magnitude

Adam

- Adam≈RMSProp+Momentum

Sequence Processing



Amazon Movie Reviews @AmznMovieRevws · Jul 24

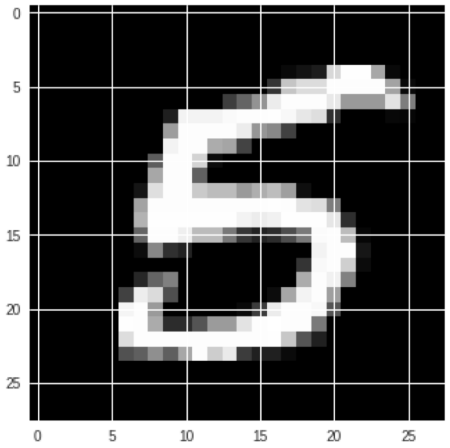
Life of Pi.

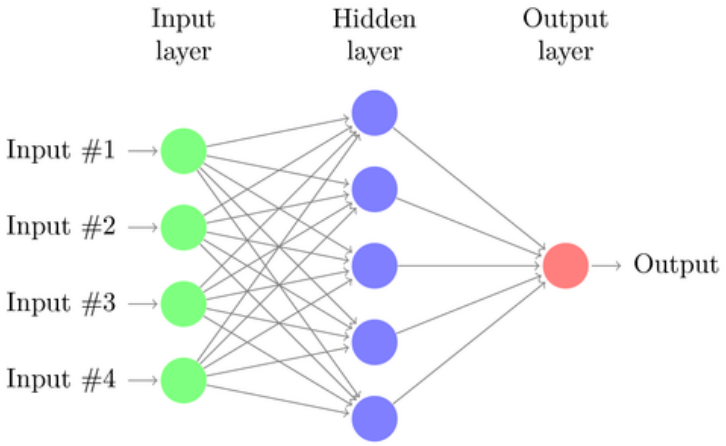
3.1415HOLY

★☆☆☆☆ **Misleading**

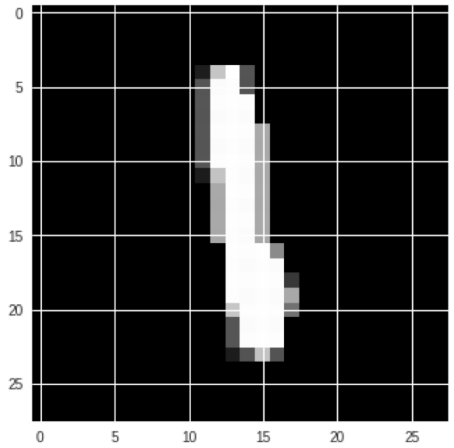
By Michael M. - May 8, 2015

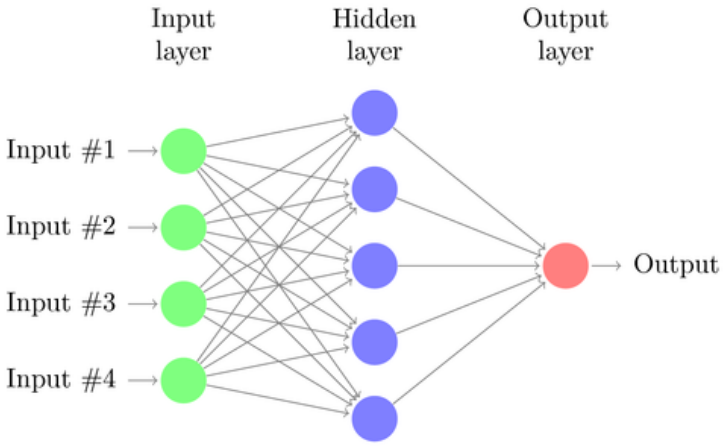
I screened this film for the students in my 9th grade algebra class and was very upset.


0
5
10
15
20
25
0
5
10
15
20
25

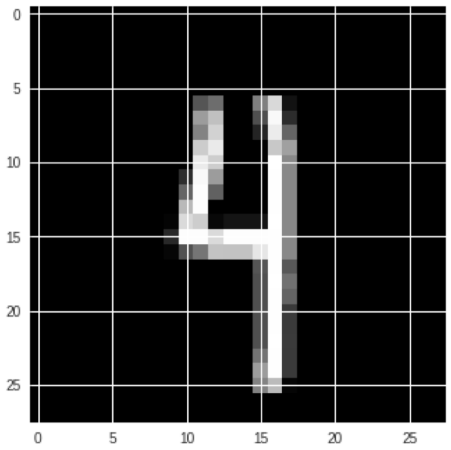

Input layer
Hidden layer
Output layer
Input #1
Input #2
Input #3
Input #4
Output

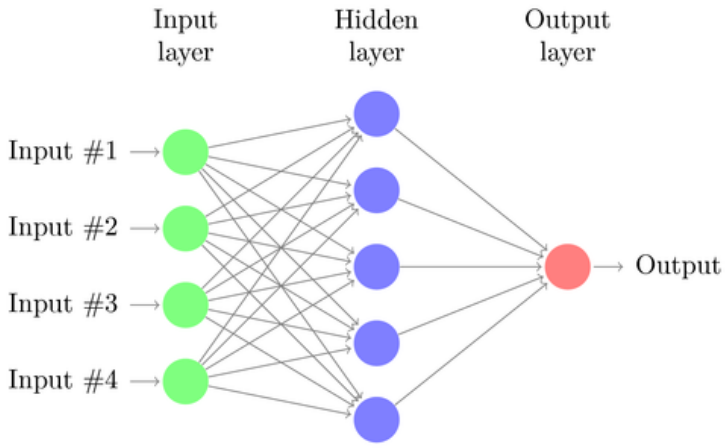
5


0
5
10
15
20
25
0
5
10
15
20
25


Input layer
Hidden layer
Output layer
Input #1
Input #2
Input #3
Input #4
Output

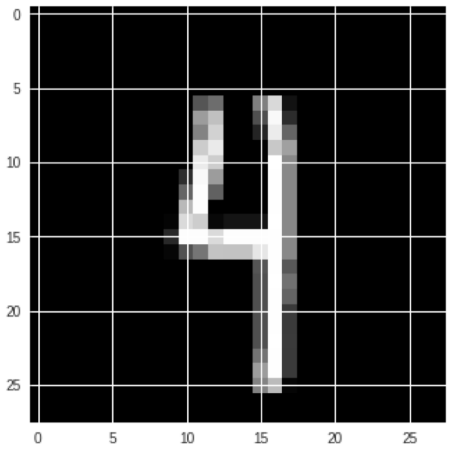
1

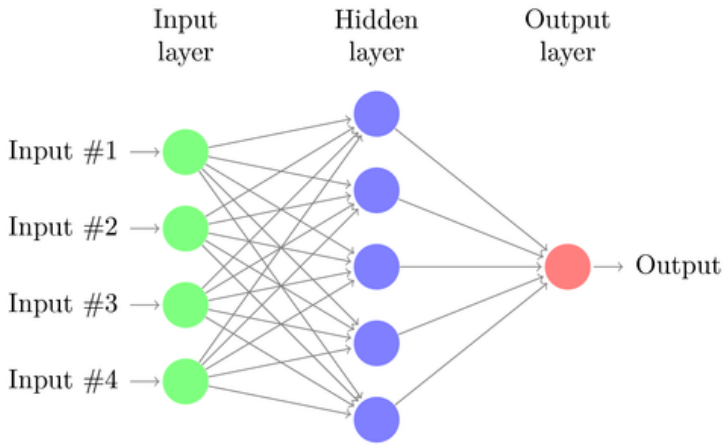

0
5
10
15
20
25
0
5
10
15
20
25


Input layer
Hidden layer
Output layer
Input #1
Input #2
Input #3
Input #4
Output

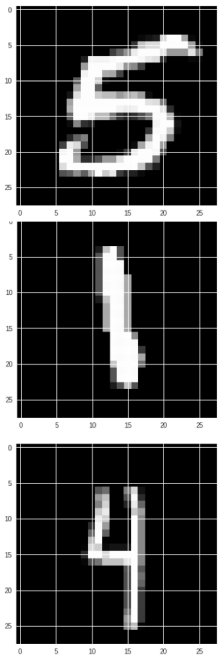
4

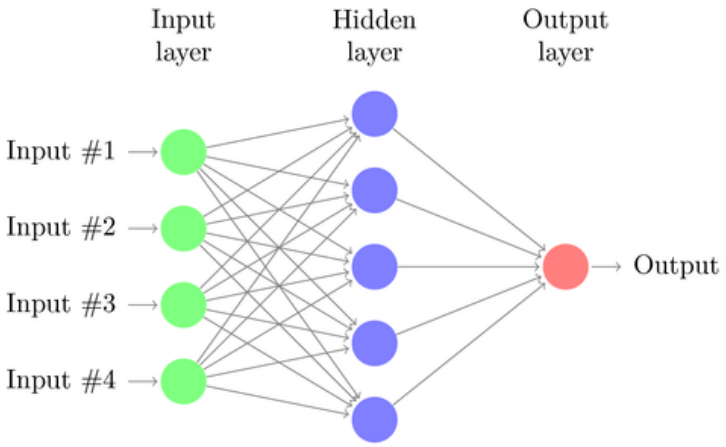
Feedforward networks have no memory





4




Input layer
Hidden layer
Output layer
Input #1
Input #2
Input #3
Input #4
Output

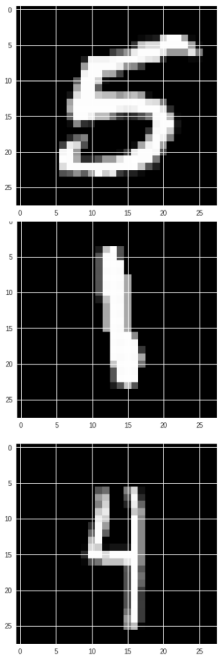
P/N?

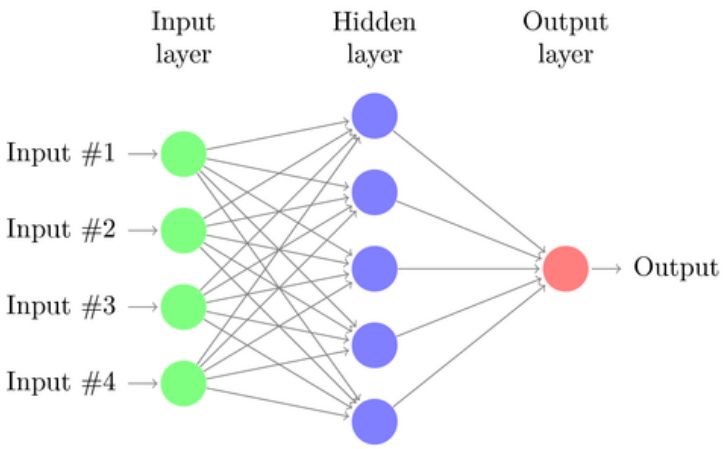
.

.

.

Not feasible for very large sequences





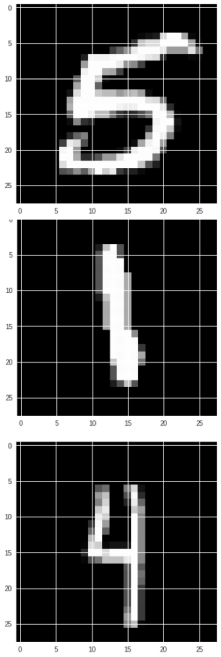
P/N?

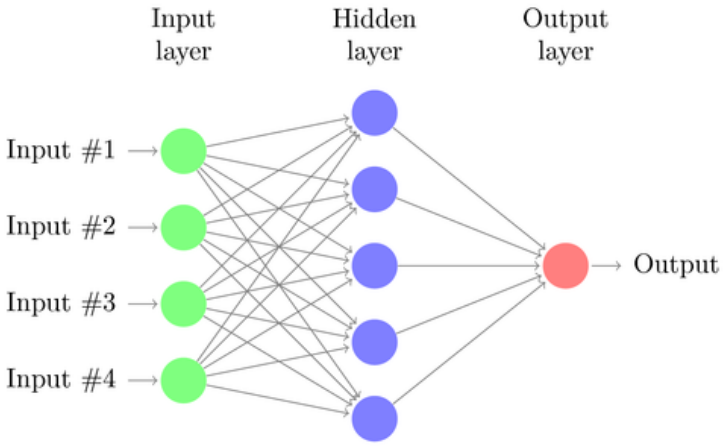
.
.
.

I went to Nepal in 2009.

In 2009, I went to Nepal.

Separate parameters for each time step.





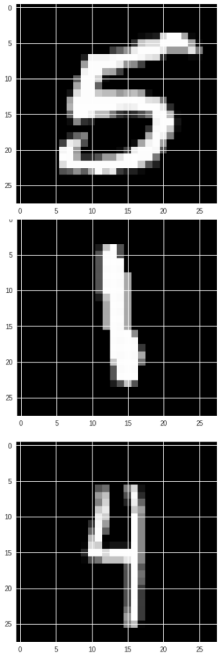
P/N?

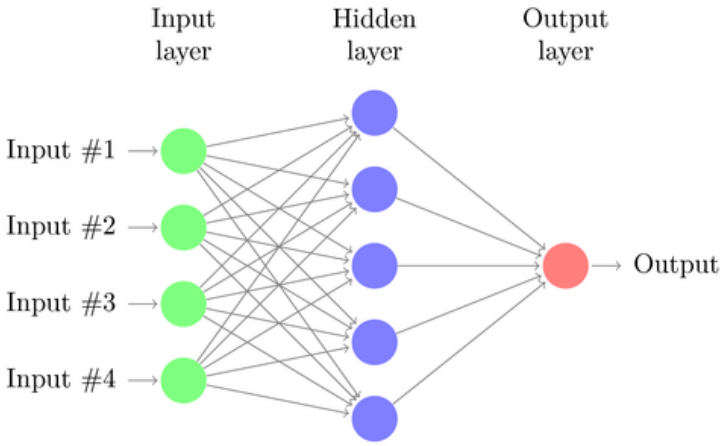
.
.
.

I went to Nepal in 2009.

In 2009, I went to Nepal.

No/poor generalization, especially for sequence sizes not observed during the training.





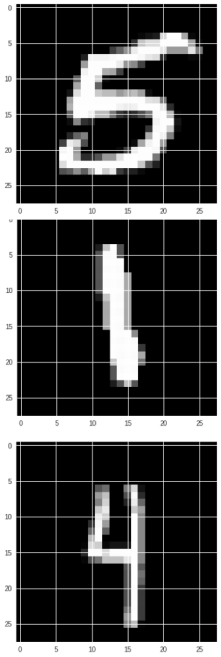
P/N?

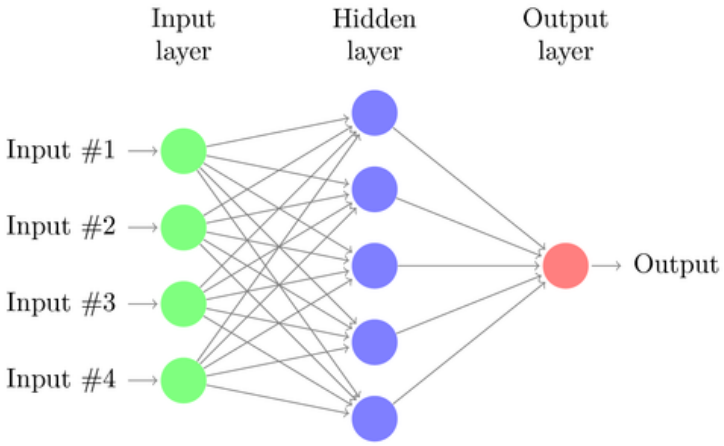
.
.
.

I went to Nepal in 2009.

In 2009, I went to Nepal.

No/poor generalization, especially for sequence sizes not observed during the training.





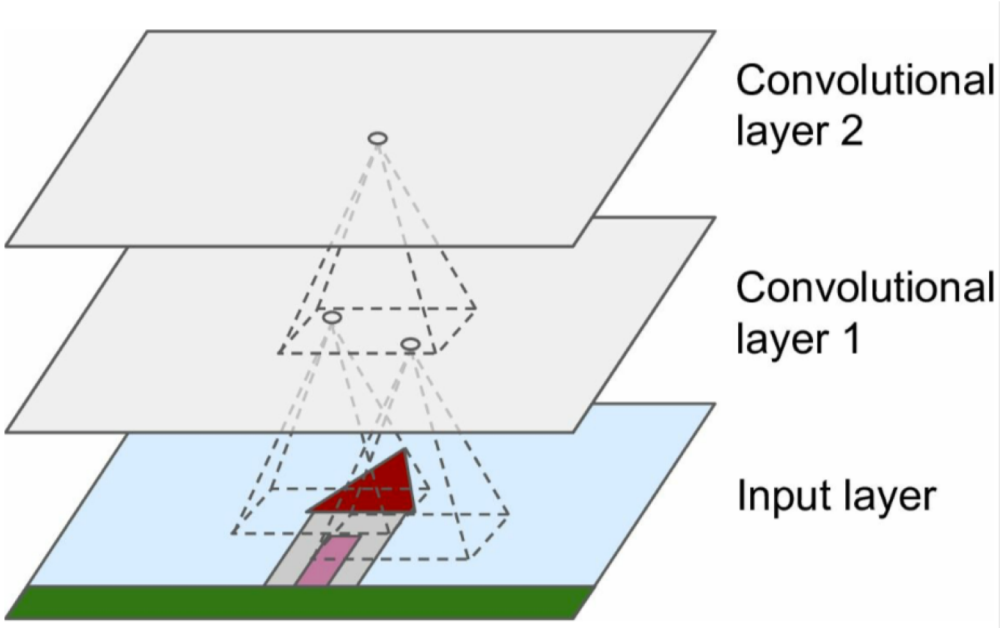
P/N?

.
.
.

When you have separate parameters for each location, all rules need to be separately learned for each position.

CNNs

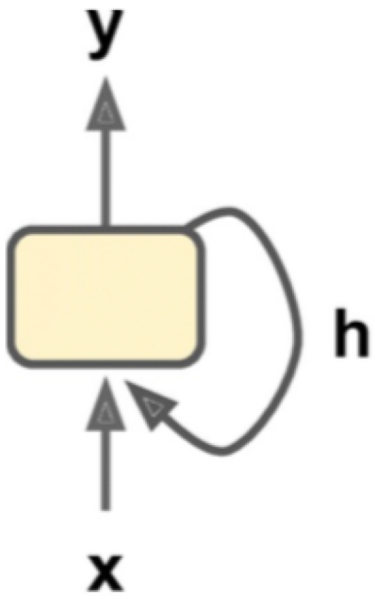
- CNNs are specialized to handle a grid of data, e.g. an images.
- And they can scale easily for larger images.
- Parameter sharing through application of kernels.

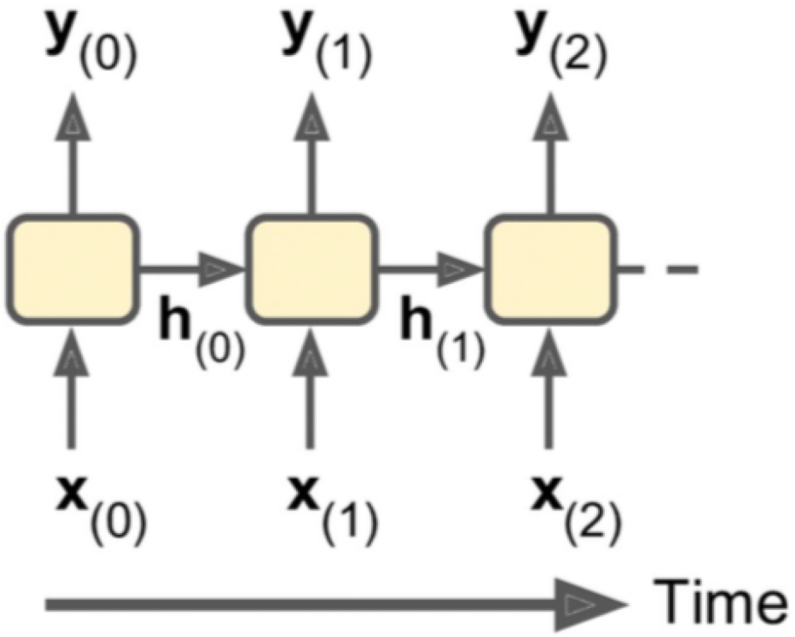


Picture from: A. Geron, Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems

Recurrent Neural Networks

(opposed to feedforward Neural Networks)





$$\hat{y} = \sigma(X.W_x + Y.W_y + b)$$