Comupational Social Science Methods

Neural Network

Taehee Kim

Summer Semester 2022

University of Oldenburg

Neural Network and Deep Learning

What is Neural Network?

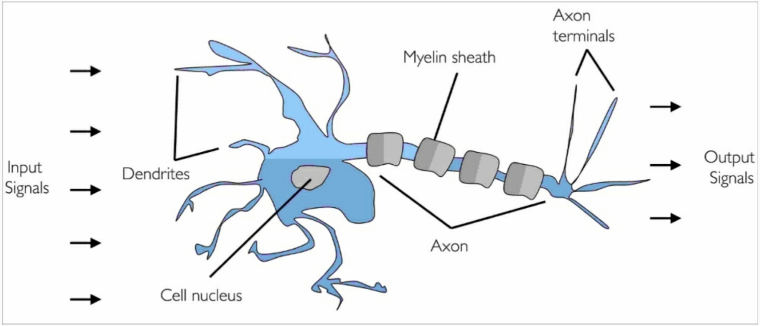


Figure 1: Neurons in brain

What is Neural Network?

Figure 1: Neurons in brain

Neural Network

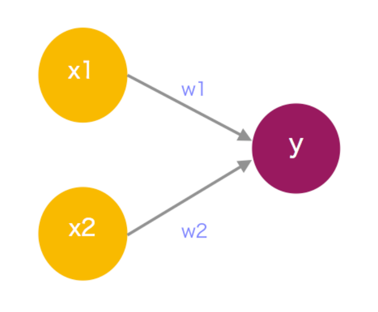
Definition

"Artificial neural networks" are massively parallel interconnected networks of simple elements and their hierarchical organizations which are intended to interact with the objects of the real world in the same way as biological nervous systems do. (Kohonen, 1988)

History

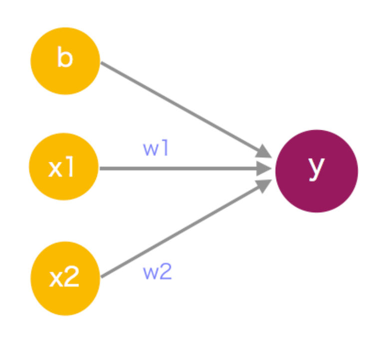
- Scientists proposed neural networks to mimic the brain
- Neural networks are widely used in 80s and early 90s, but diminished in late 90s
- With the increasing of computational power, neural networks become resurgent

Logic of Neural Network: Perceptron



$$y = \begin{cases} 0 \text{ if } x_1 w_1 + x_2 w_2 \leq \theta \\ 1 \text{ if } x_1 w_1 + x_2 w_2 > \theta \end{cases}$$

Logic of Neural Network: Perceptron

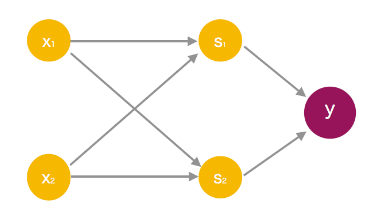


$$y = \begin{cases} 0 \text{ if } b + x_1 w_1 + x_2 w_2 \leq 0 \\ 1 \text{ if } b + x_1 w_1 + x_2 w_2 > 0 \end{cases}$$

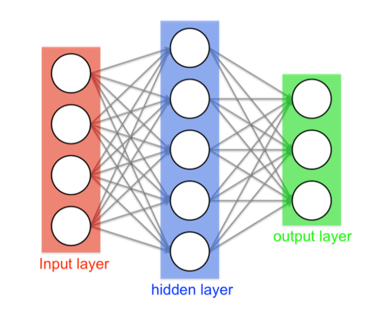
Express Truth Table using Perceptron

Input		Output			
X0	X1	AND	OR	NAND	XOR
0	0	0	0	1	0
1	0	0	1	1	1
0	1	0	1	1	1
1	1	1	1	0	0

- Let's think about parameter of Perceptron (b, w1, w2) to generate each output.
- What about $X_0 \wedge X_1$? (AND)
- Can you express $X_0 \oplus X_1$? (XOR) It is a non-linear problem.



- We can express $X_1 \oplus X_2$ using multi-layered perceptron!
- while a single neuron can be regarded as a binary linear classifier, put multiple neurons together is able to apply nonlinear transformation.



- 3-layered neural network, fully-connected
- Input, hidden, and output layer
- How many parameters?

- $5 + 3 =$ neurons
- $5 \times 4 + 3 \times 5 = 35$ and $5 + 3 = 8$ biases, 43 parameters in total.

- Modern CNNs contains of 100 million parameters with 10-20 layers $\rightarrow$ deep learning

Activation function

- defines the output of the neuron given set of inputs

  $a = b + x_1 w_1 + x_2 w_2$

  $y = h(a) \rightarrow$ activation function h()

Popular activation function

- sigmoid function: $h(x) = \frac{1}{1+exp(-x)}$
- ReLU (rectified Linear Unit): $h(x) = max(x, 0) \rightarrow$ works well with non-linear case

Activation function in output layer

- choose based on the problem at stake: classification or regression?
- classification: use Softmax function
- regression: use identity function

Design of Neural Network

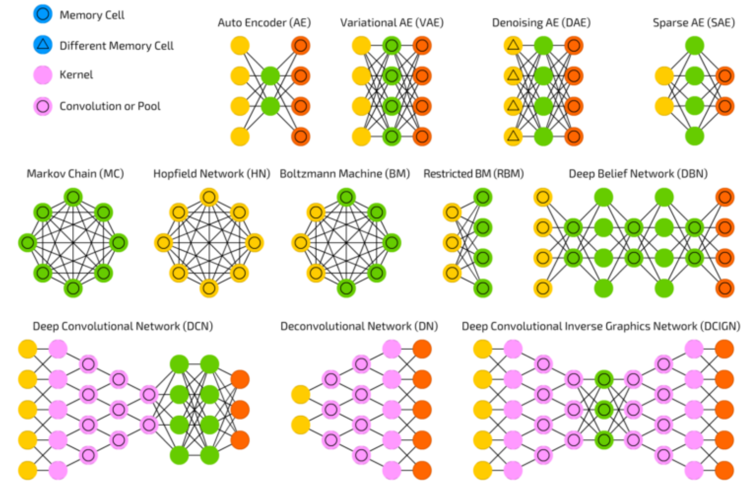


Figure 2:

Example of hand written number classification (MINST)

- Number of inputs: number of pixels (if an image is 28×28 gray color $= 784$)
- Number of outputs: 10 (since there are 10 categories)

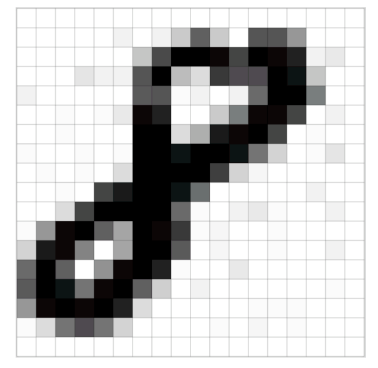


Figure 3: source: from here by Adam Geitgey

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	12	0	11	39	137	37	0	152	147	84	0	0	0
0	0	1	0	0	0	41	160	250	255	235	162	255	238	206	11	13	0
0	0	0	16	9	9	150	251	45	21	184	159	154	255	233	40	0	0
10	0	0	0	0	0	145	146	3	10	0	11	124	253	255	107	0	0
0	0	3	0	4	15	236	216	0	0	38	109	247	240	169	0	11	0
1	0	2	0	0	0	253	253	23	62	224	241	255	164	0	5	0	0
6	0	0	4	0	3	252	250	228	255	255	234	112	28	0	2	17	0
0	2	1	4	0	21	255	253	251	255	172	31	8	0	1	0	0	0
0	0	4	0	163	225	251	255	229	120	0	0	0	0	0	11	0	0
0	0	21	162	255	255	254	255	126	6	0	10	14	6	0	0	9	0
3	79	242	255	141	66	255	245	189	7	8	0	0	5	0	0	0	0
26	221	237	98	0	67	251	255	144	0	8	0	0	7	0	0	11	0
125	255	141	0	87	244	255	208	3	0	0	13	0	1	0	1	0	0
145	248	228	116	235	255	141	34	0	11	0	1	0	0	0	1	3	0
85	237	253	246	255	210	21	1	0	1	0	0	6	2	4	0	0	0
6	23	112	157	114	32	0	0	0	0	2	0	8	0	7	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 3: source: from here by Adam Geitgey

Convolutional Neural Network

Convolutional Neural Network[1]

Convolutional Neural Network (CNN)

- suitable to handle image data
- Basic procedure of the method is the same as ordinal Neural Network method (i.e., CNNs create neurons which have weights and biases obtained from learning, and utilize a loss function in the last layer)
- The idea of CNNs is that divide an image into small area of images (pixels) and compress their information, i.e., “convolution".

If we use ordinal neural network to classify image data..

- every pixels are considered independently
  - results are sensitives to small differences in images
  - large number of weights (if we have normal size of images such as 350x350 = 122,500 inputs)

[1] Recommended reading: the lecture note of the course “CS231n: Convolutional Neural Networks for Visual Recognition" in the Stanford University. here

Three types of layers in CNN

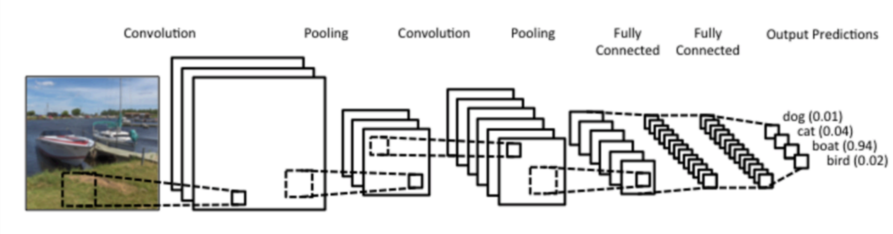


Figure 4: source: from here by Rohith Gandhi

Three types of layers in CNN

- Convolution, Pooling, Fully connected

Common pattern of CNN

$$INPUT \rightarrow ((Conv. \rightarrow RELU) \times N \rightarrow Pooling) \times M \rightarrow (\mathrm{FC} \rightarrow \mathrm{RELU}) \times K \rightarrow FC$$

, where $0 < N \leq 3$, $M > 0$, $0 \leq K \leq 2$

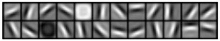


Figure 5: First layer[2]

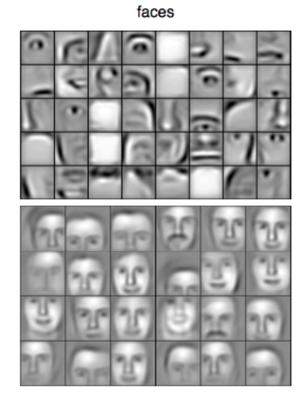


Figure 6: Second, third layer

[2]source: Lee et al., 2009, "Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations", Proceedings of the 26th International Conference on Machine Learning, Montreal, Canada.

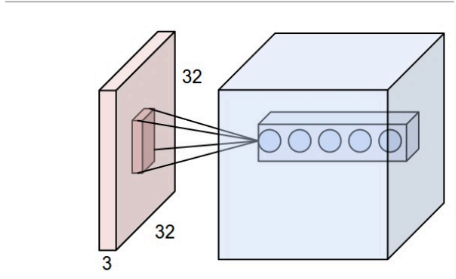


Figure 7: source: CS231n in the Sanford Univ. here

Convolution

- compress small area of images by using a filter (or kernel)
- each filter across the width and height of the input volume and compute dot products between the entries of the filter (=weights) and the input at any position

  $\rightarrow$ it produces a 2-dimensional activation map that gives the responses of that filter at every spatial position
- Use activation function RELU

Parameter we need to set:

- number of the layer
- size of the filter: 5x5 or 3x3 is common
- stride of the filter: 1 or 2 is common
- the size of zero-pedding

The zero–pedding

- add zero values around the image
- allow us to control the size of the output
- we can increase coverage of the edge area

The size of the output

- $(W - F + 2P)/S + 1$, W: the input size, F: filter size, P: pedding size, S: stride
- Example: the input size is 48 (48*48), filter size is 5 (5*5), and stride is 1. We want to have the same size of the output as the input size. What should be P here?
- $\rightarrow$ 2.

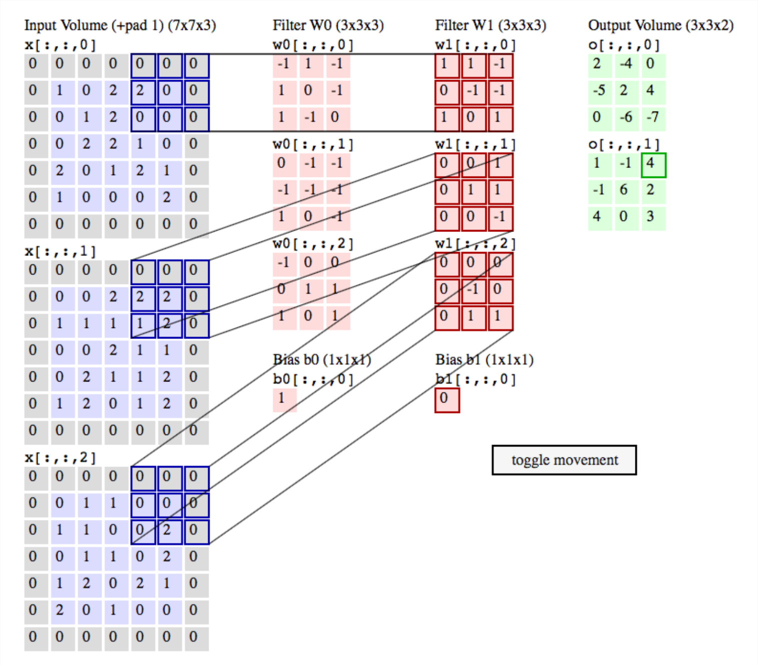
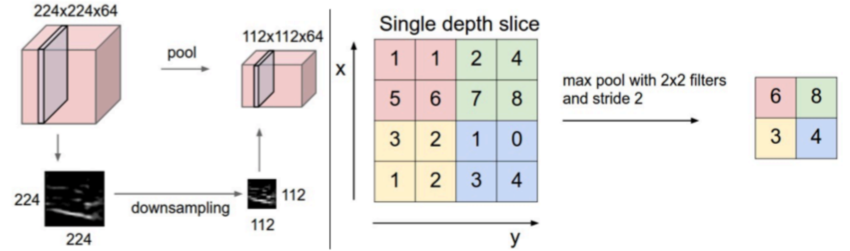


Figure 8: source: CS231n in the Sanford Univ. here



Pooling layer downsamples the volume spatially, independently in each depth slice of the input volume. **Left:** In this example, the input volume of size [224x224x64] is pooled with filter size 2, stride 2 into output volume of size [112x112x64]. Notice that the volume depth is preserved. **Right:** The most common downsampling operation is max, giving rise to **max pooling**, here shown with a stride of 2. That is, each max is taken over 4 numbers (little 2x2 square).

Figure 9:

- Pooling Layer reduce the size of the previous input, so that we can reduce the number of parameter $\rightarrow$ reduce the risk of overfitting
- Max pooling that choose maximum value of the area is commonly used.

  Example: if we choose 2*2 pooling from 32*32 input, new input would be

Fully connected layer

- all neurons in previous layers are connected. All activations are computed here and produce prediction by using cost function (such as SVM or Softmax)

Hyperparameters

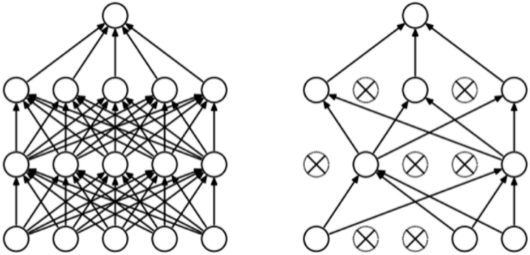
Epoch

- An epoch is a single step in training a neural network (i.e., one forward/backward)
- train multiple times (use multiple epochs) help to find optimal parameters
- too many times of epochs $\rightarrow$ could lead to overfitting

Batch size

- batch size is the number of training examples in one forward/backward train.
- The higher size of batch: use more memory space
- The lower size of batch: use less memory, fast, but less accurate

Dropout: 'dropout' some of neurons which is chosen at random during the certain training pass (forward/backward) in order to reduce the risk of overfitting.



(a) Standard Neural Net (b) After applying dropout.

Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting", JMLR 2014

TensorFlow and Keras

TensorFlow

- ..is a scarable and multiplatform programming interface for implementing and running ML, including convenience wrappers for deep learning
- one of the most popular deep learning libraries currently available
- it can let us implement neural networks much more efficiently $\rightarrow$ speed up ML calculation significantly, allow us to utilize GPUs

Keras

- Keras is libraries that is built on top of TensorFlow
- it allow us to implement neural networks in only a few lines of code in very intuitive way

We install TensorFlow and Keras.

- Open Anaconda Prompt or Terminal
- `conda create -n tensorflow_env python=3.5`
  → create an environment with python 3.5 which name is 'tensorflow_env'
- Windows: `conda activate tensorflow_env`
  Mac: `source activate tensorflow_env`
  → activate the environment
- `conda install -c conda-forge tensorflow`
  → install tensorflow in the environment
- `conda install -c conda-forge keras`

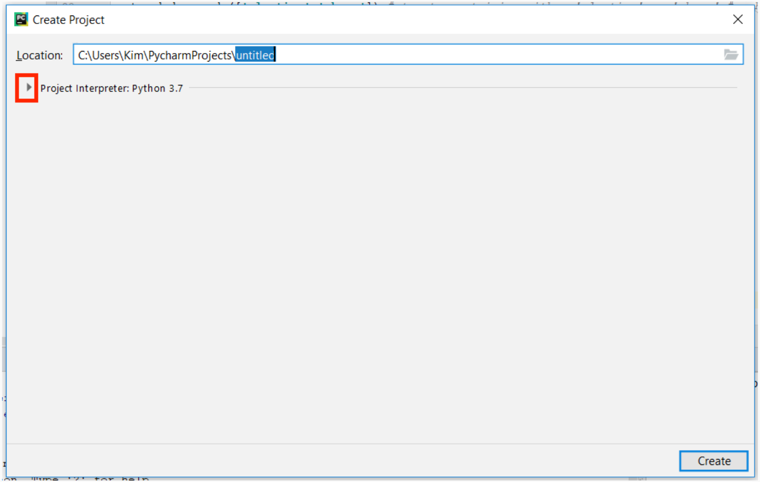
Check conda environment

- `conda env list`

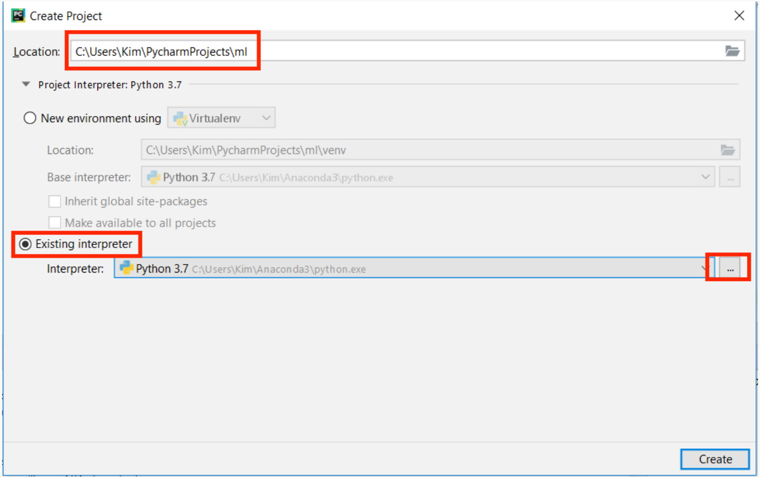
```
# conda environments:
#
base                     C:\Users\Kim\Anaconda3
tensorflow_env        *  C:\Users\Kim\Anaconda3\envs\tensorflow_env
```

Remember the address of the new environment.

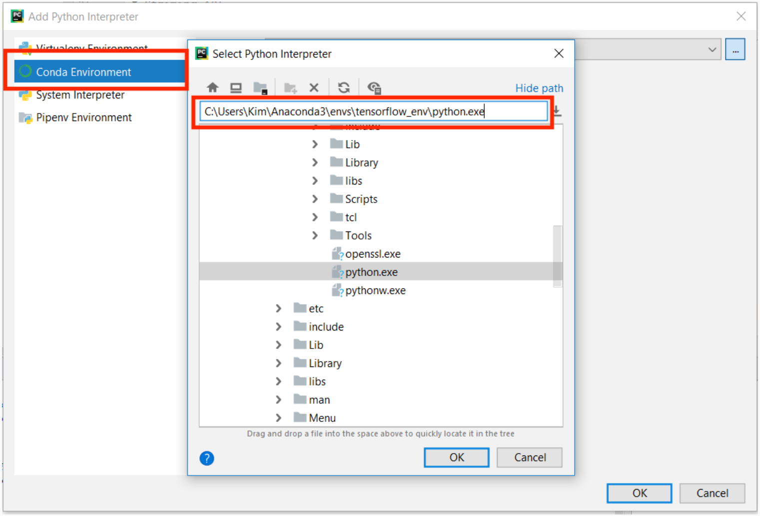
Add new interpreter in Pycharm



Add new interpreter in Pycharm



Add new interpreter in Pycharm



Add new interpreter in Pycharm