Deep Learning
Image Classification

BMI701 Introduction of Biomedical Informatics
Lab Session 10

Wei-Hung Weng

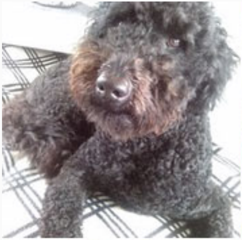
November 30, 2016

HMS DBMI — MGH LCS

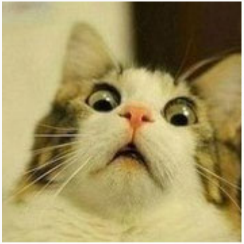
Deep Learning?

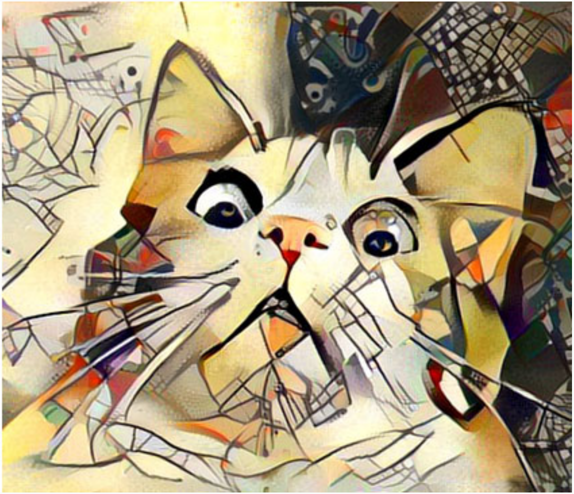
- As Go master
- As an artist
- Deep style
- Deep dream
- How DNN think about your selfie
- Deep learning Google trend
- Simply to say, less feature engineering by hand
- Multiple layers perceptron
- IDEA! Rebrand Neural Nets $\rightarrow$ Deep Nets
- $\uparrow$ Computational power
- Very good Nature review paper by three big names (LeCun, Bengio, Hinton)

Vincent Van Dog



Kandinsky Cat





Outline

- Brief introduction of deep learning
- Algorithms
- Using R to do image classification

Why Deep Learning?

- Shallow vs. deep?
  - Shallow network can represent any function (given enough hidden neurons), but deep structure is more effective
  - Less parameteres
  - Less training data and hand-crafted features
  - More machine learning
- Modularization

Step by Step

- Network Structure: A set of function
- Learning Target: Define the goodness of a function
- Learn: Pick the best function f

Network Structure

- Neuron
- Input layer
- Hidden layers
- Output layer (softmax)
- Weight / bias $\rightarrow$ network parameters θ
- Activation function: sigmoid / ReLU
- Fully connected feedforward network

Learning Target?

- Make your **loss** as small as possible
- Find a function f and parameters θ to
- $argmin L = \sum_{r=1}^{R} l_r$

Learn

- Gradient descent
- Randomly pick up w
- Compute $\partial L / \partial w$
  - Negative $\rightarrow$ increase w
  - Positive $\rightarrow$ decrease w
  - $w \leftarrow w - \eta \partial L / \partial w$
  - η is so called learning rate
- Repeat and repeat until $\partial L / \partial w \rightarrow 0$
- Do gradient decent on all w and $b \rightarrow \nabla L$
- Sometimes it goes to local minimum when you assign different initial w
- Using backpropagation to compute $\partial L / \partial w$
- Don't use your hand, use the developed tools...
  - Theano, Tensorflow, Caffe, Torch, ...
  - Keras

Some Tips

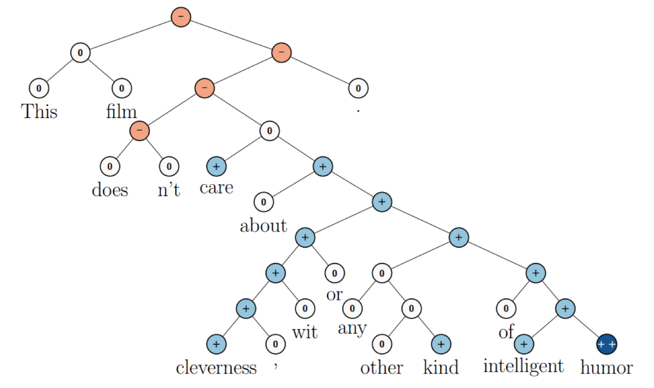
- Choose proper loss: square error vs. cross entropy
  - Use CE when you use softmax output
- Mini-batch
  - Update once after going through all sample
  - Update 100 times in one epoch if you have 100 batches
  - Fast convergence
- New activation function
  - When you have too many layers
  - GD will almost disappear, learn very slow and almost randomly
  - Use ReLU / maxout instead of sigmoid
- Adaptive learning rate
  - Adagrad ($\eta_w = \frac{\eta}{\sqrt{\sum_{i=0}^{t}(g^i)^2}}$)
- Momentum
  - Add some momentum to get out of local minimum
  - Adam algorithm (Advanced Adagrad + Momentum)

Some Tips

- Early stopping
- Regularization / weight decay
  - $w \leftarrow$ smaller and smaller...$w - \eta \partial L / \partial w$
- Dropout
  - Remove $x\%$ neurons in each mini-batch (resampling)
  - Network structure will be different
  - Do better job?
  - Sort of emsemble method (different batch, different network, average them)
- Different network structures
- Deep Playground by Google Tensorflow

Network Structure

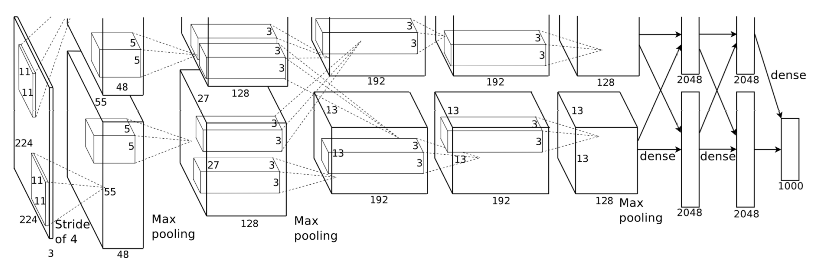
- RNN / LSTM



Socher 2012

Network Structure

- CNN
  - Convolution → max pooling → convolution → max pooling → ... → flatten → fully connected feedforward → softmax



Krizhevsky 2012

Why CNN?

- Some patterns are much smaller than the whole image: No need to use whole image as the input of the neuron
- Some patterns appear again and again in different places: Neurons do the almost same thing so they can share the same parameters
    - Convolution
- Subsampling will not change your perception
    - Max pooling
- So in fact, CNN is a simpler ANN architecture with less parameters
- But your input should have the features related to image recognition
    - AlphaGo use 5×5 for the first layer
    - But AlphaGo didn't do subsampling (no max pooling)

Network Structure of Alpha Go

Neural network architecture. The input to the policy network is a $19 \times 19 \times 48$ image stack consisting of 48 feature planes. The first hidden layer zero pads the input into a 23×23 image, then convolves k filters of kernel size 5×5 with stride 1 with the input image and applies a rectifier nonlinearity. Each of the subsequent hidden layers 2 to 12 zero pads the respective previous hidden layer into a 21×21 image, then convolves k filters of kernel size 3×3 with stride 1, again followed by a rectifier nonlinearity. The final layer convolves 1 filter of kernel size 1×1 with stride 1, with a different bias for each position, and applies a softmax function. The match version of AlphaGo used $k = 192$ filters; Fig. 2b and Extended Data Table 3 additionally show the results of training with $k = 128$, 256 and 384 filters.

Silver, Huang 2016

What Does CNN Learn?

- 1st filter: Easy to understand
- 2nd filter: Input is the value after convulition and max pooling, NOT pixels!
- Degree of the activation of the k-th filter: use gradient ascent
- Filter before flatten: detect texture
- Filter after flatten: larger scale patterns
- But... DNN are easily fooled
  - Use regularization (e.g. LASSO)

Imaging Classification

- Using Theano
  - Courtesy by DeepLearning.net
  - Theano documentation
- Using H2O in R

Some Deep Learning Resource

- TED talk: Fei-Fei Li
- Theorerical deep learning: Yoshua Bengio
- Udacity: Deep learning course
- Stanford CS231n: CNN for Visual Recognition (Li)
- Stanford CS224d: Deep Learning for NLP (Socher)
- Deep Learning Book

Summary

- Thank you for coming to my lab sessions!
- Contact
    - Github repository
    - ckbjimmy@gmail.com
    - Linkedin: Wei-Hung Weng