Big Bio-Data Analysis (Artificial Intelligence and Machine Learning)

22 July 2022

Introduction to Deep Learning

By

Richard Sserunjogi
Department of Computer Science,
Makerere University, Uganda
sserurich@gmail.com

AFRICAN CENTERS OF EXCELLENCE IN BIOINFORMATICS & DATA INTENSIVE SCIENCE

What is Deep Learning?

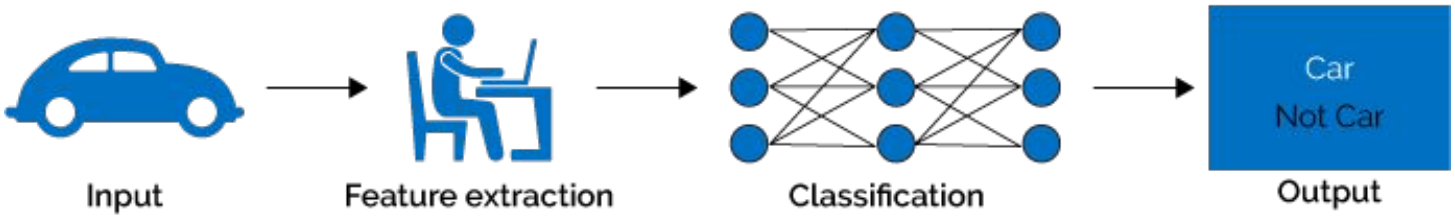
Deep learning is a subset of machine learning where algorithms inspired by the human brain(neural networks) learn from large amounts of data.

Good old Neural Networks with more layers

Essentially any neural network with two or more hidden layers is deep.

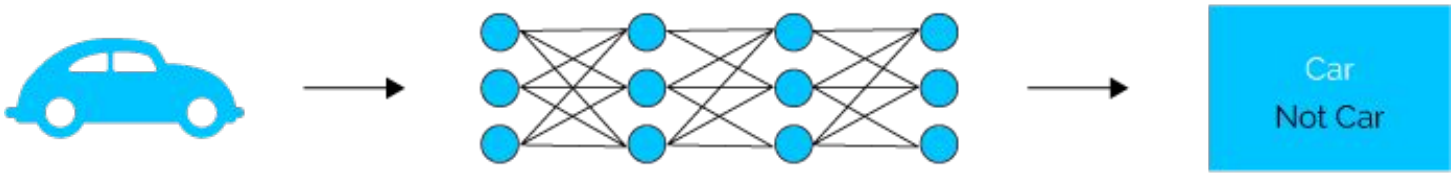
Deep learning algorithms attempt to learn (multiple levels of) representation by using a **hierarchy of multiple layers**

Machine Learning



Input → Feature extraction → Classification → Output

Deep Learning



Input → Feature extraction + Classification → Output

Feature Engineering vs Learning

- Feature engineering is the process of using domain knowledge of the data to create features that make machine learning algorithms work.
- “When working on a machine learning problem, feature engineering is manually designing what the input x's should be.”

  -- Shayne Miel
- “Coming up with features is difficult, time consuming, requires expert knowledge.”

  -- Andrew Ng

Why Deep Learning Now?

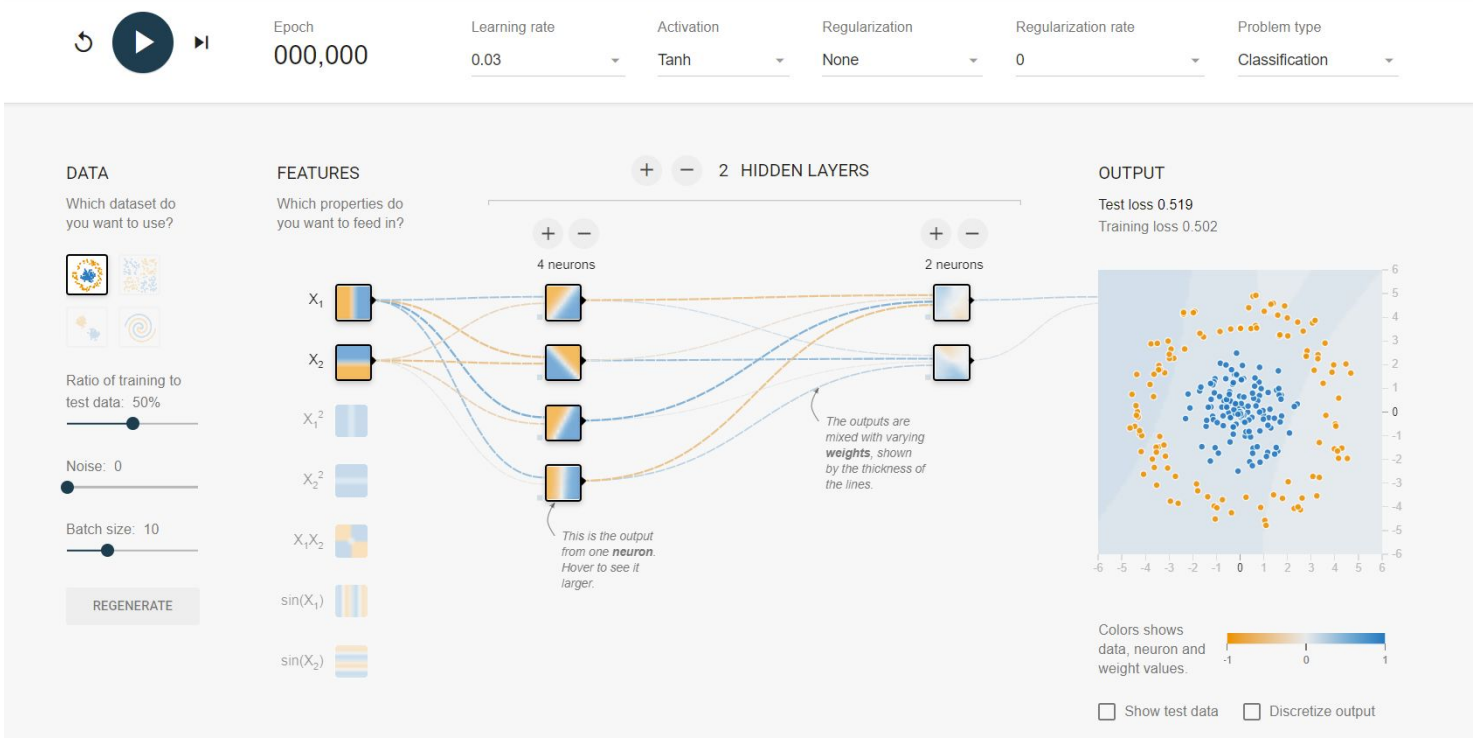
- Hardware Improvement (GPU, TPU)
- Data (labeled data)
- Algorithmic advances (*activation functions* for neural layers, better Optimisation algorithms)
- new wave of investment
- Open source tools and libraries

Deep learning Applications

- Natural Language Processing
  - Machine Translation, Speech Recognition
- Computer Vision
  - Self driving cars, health care , drug discovery
- Games : Chess and Go
- Robotics
- Biological sequences
  - Predicting DNA motifs or protein classes
  - Protein folding

Tools for Deep Learning

LIVE DEMO



[link to demo]

Deep Learning Models

Common deep learning models:

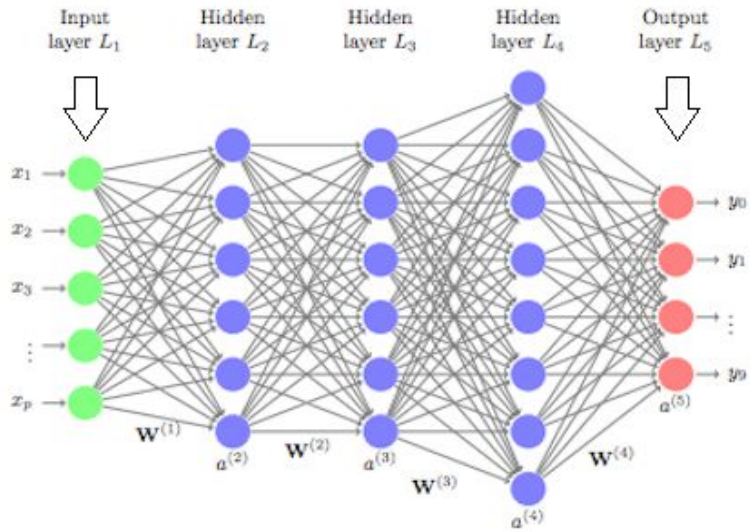
- Deep Feed Forward Neural Networks (DNN)
- Convolutional Neural Networks(CNNs)
- Recurrent Neural Networks (RNNs)
- Long Short Term Memory Neural Networks (LSTMs)

Others

- Generative Adversarial Networks (GANs)
- Encoder Decoder model

Deep Feed Forward Neural Networks

Feed forward neural networks with more than 2 hidden layers



Convolutional Neural Networks

Class of deep neural networks, most commonly applied to analyzing visual imagery

CNNs have been so successful at computer vision tasks

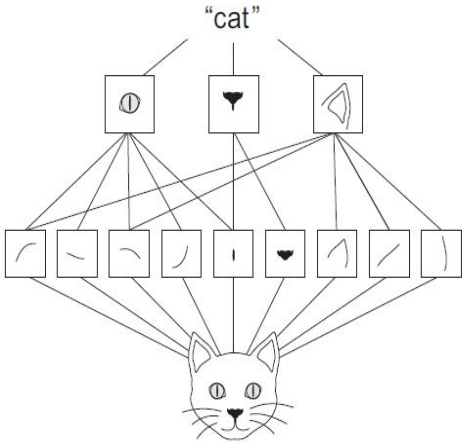
Key two characteristic /properties of CNNs:

- *They can learn spatial hierarchies of patterns*
- *The patterns they learn are translation invariant.*

Convolution Operation

Convolutions operate over 3D tensors, called ***feature maps*****,** with two spatial axes (*height* and *width*) as well as a *depth* axis (also called the *channels* axis).

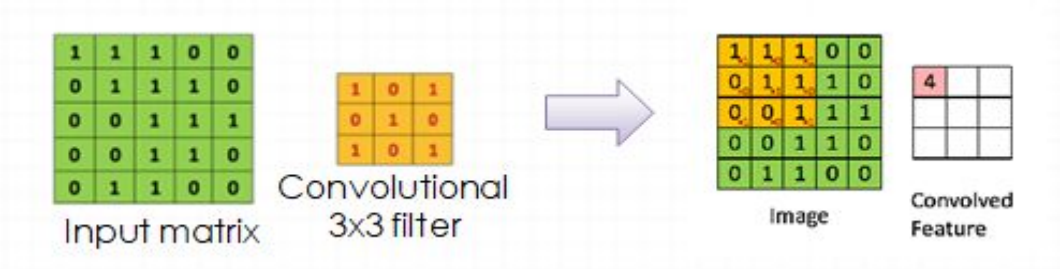
Convolution operation extracts patches from its input feature map and applies the same transformation to all of these patches, producing an *output feature map*



Convolutions are defined by two key parameters:

- *Size of the patches extracted from the inputs*
- *Depth of the output feature map*

These are typically 3 × 3 or 5 × 5. In the example, they are 3 × 3, which is a common choice.



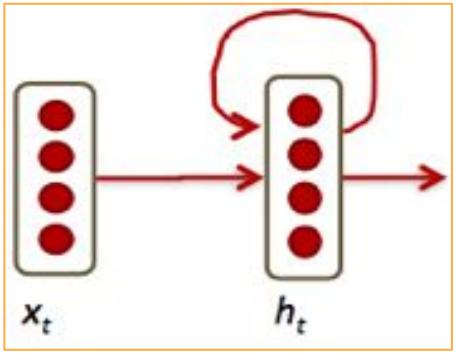
Stride: *Number of steps a filter is moved*

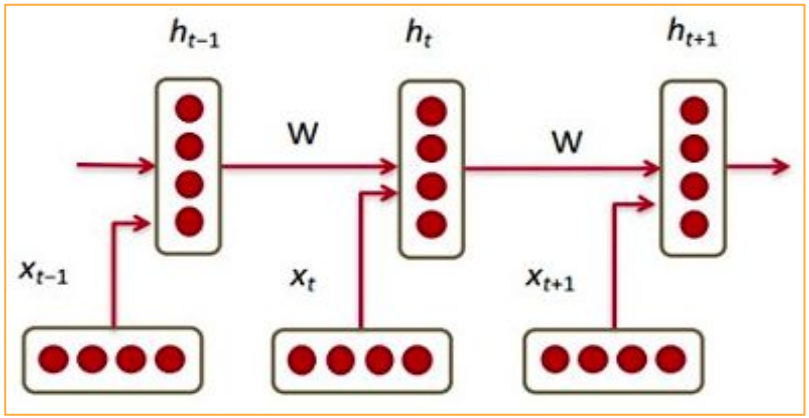
Recurrent Neural Networks

Humans don’t start their thinking from scratch every second.

RNN address this issue.

They are networks with loops in them, allowing information to persist.





$$h_t = \sigma(W^{(hh)} h_{t-1} + W^{(hx)} x_t)$$

RNNs can be used for sequential prediction and time series prediction.

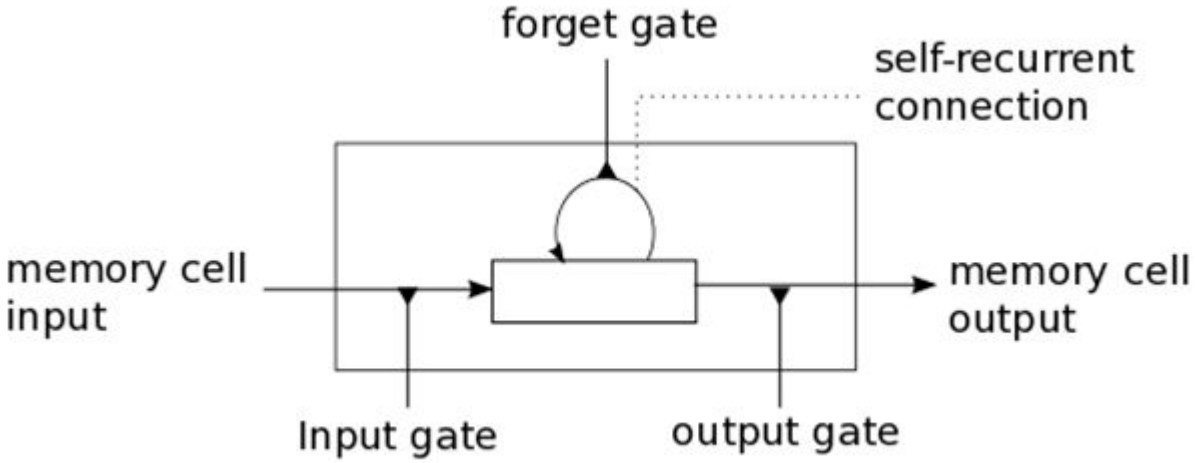
RNNs experience the problem of Long-Term Dependencies and this is solved by a special kind of RNN called LSTMs

Long Short Term Memory Neural Networks

LSTMs are a special kind of RNN capable of learning long-term dependencies.

They have some internal contextual state cells that act as long-term or short-term memory cells.

The output of the LSTM network is modulated by the state of these cells.



In addition to the cells, the LSTM units have gates (input gate, output gate and forget gate) that are used to control the flow of information into and out of the cells.

LSTMs elegantly address the problem of ***vanishing*** and *exploding* gradients encountered in vanilla RNNs

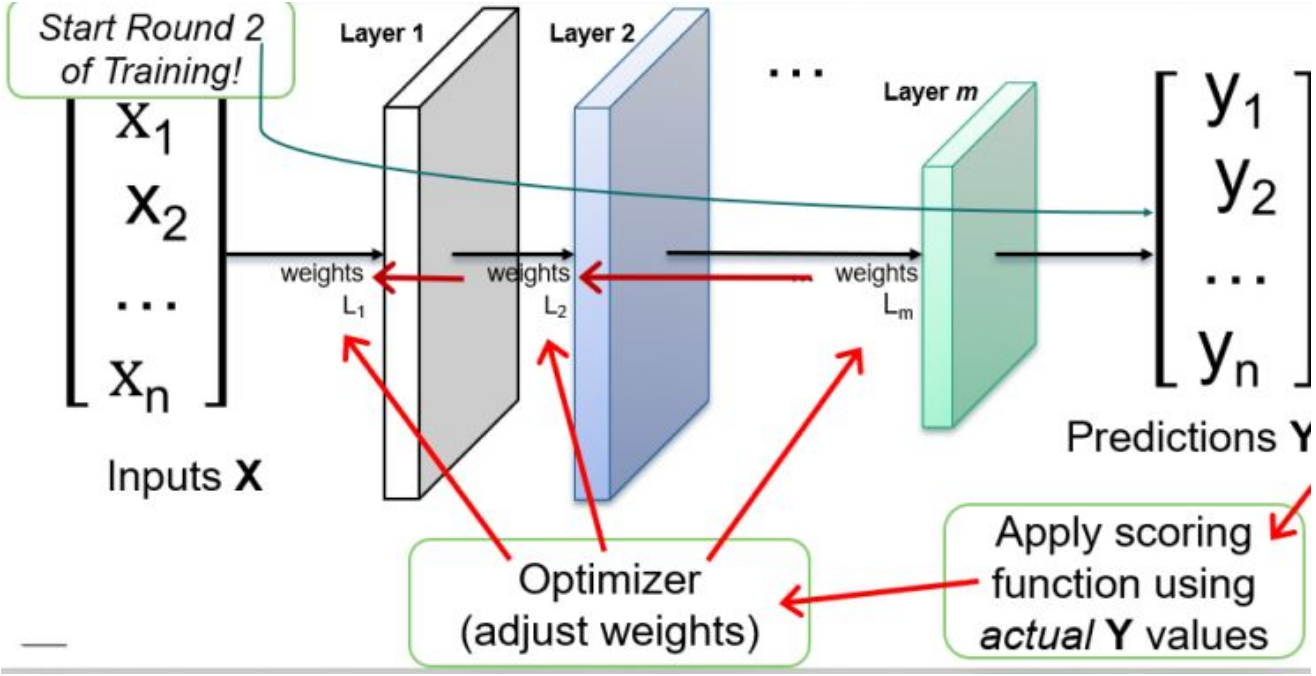
Vanishing Gradient Problem

- Difficulty found in training neural networks with gradient-based learning methods and back propagation.
- Each of the neural network's weights receives an update proportional to the partial derivative of the error function with respect to the current weight in each iteration of training.
- The problem is that in some cases, the gradient will be vanishingly small, effectively preventing the weight from changing its value.
- In the worst case, this may completely stop the neural network from further training.
-

Training Deep Neural Network

- How do DNNs learn so well? (Backpropagation)
  - The key is they compute answers across layers in a forward pass and then to use a backwards pass to optimize the weights.
  - This way all layers are updated each round (sometimes called an ‘epoch’) of training!
- How does this backward pass work?
  - The chain rule (remember from calculus?) – where we can calculate the derivative (the slope or rate of change) from two or more functions.

Training Deep Neural Networks (Cont'd)



Training Deep Neural Networks (Cont’d)

- Given an example (or group of examples)
    - we know how to compute the derivative for each weight.
- How exactly do we update the weights?
- How often? (after each training data point? after all the training data points?)

Gradient Descent

W_new= W_old- lr* derivative

Classical approach—get derivative for entire data set, then take a step in that direction

Pros: Each step is informed by all the data

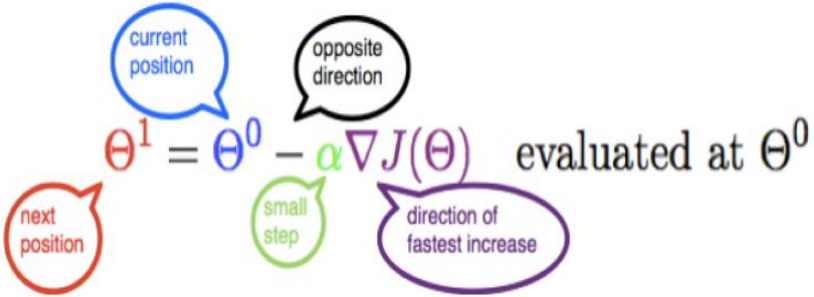
Cons: Very slow, especially as data gets big

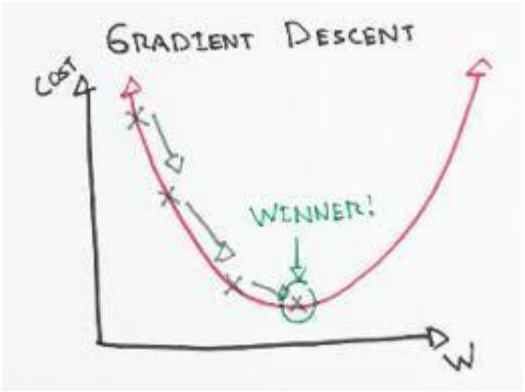
Gradient Descent

Step 1: Calculate the gradient of the function w.r.t the parameter. If there are multiple parameters, then use partial derivatives w.r.t the various parameters.

Step 2: Calculate the descent value for different parameters by multiplying the value of derivatives with learning or descent rate (step size) and -1.

Step 3: Update the value of parameter by adding up the existing value of parameter and the descent value.





Another Approach: Stochastic Gradient Descent

- Get derivative for just one point, and take a step in that direction
- Steps are “less informed” but you take more of them
- Should “balance out”
- Probably want a smaller step size
- Also helps “regularize”

Compromise approach: Mini-batch

Compromise approach: Mini-batch

Get derivative for a ”small” set of points, then take a step in that direction

Typical mini batch sizes are 16, 32

Strikes a balance between two extremes

Compromise approach: Mini-batch

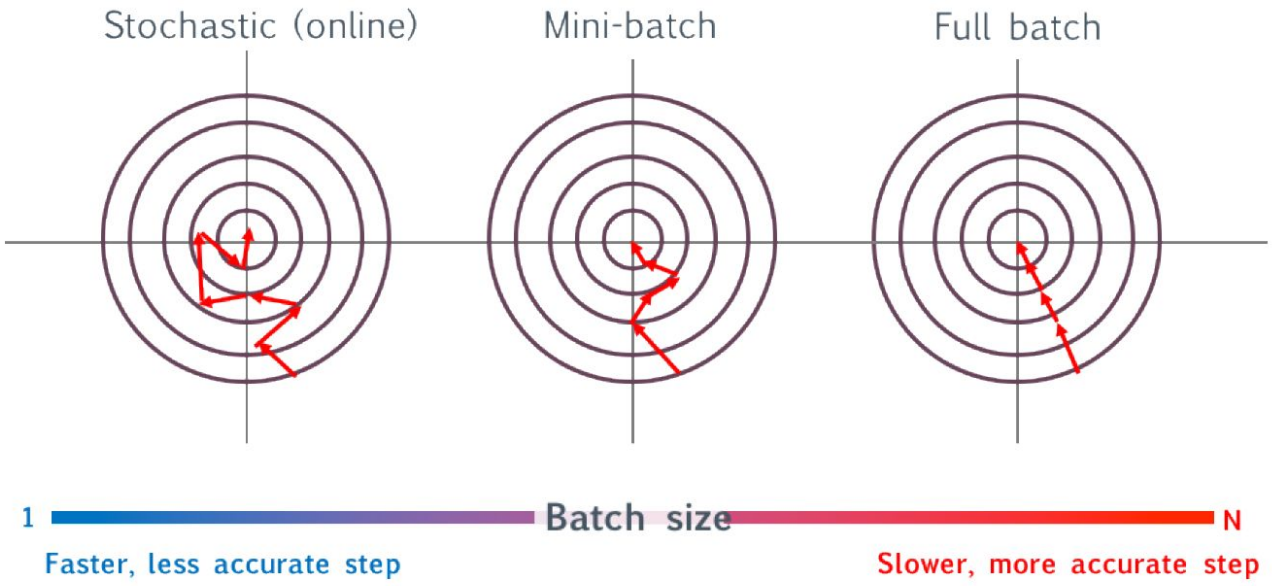
Full-batch: Use entire data set to compute gradient before updating

Mini-batch: Use a smaller portion of data (but more than single example) to compute gradient before updating

Stochastic Gradient Descent (SGD): Use a single example to compute gradient before updating

An Epoch : refers to a single pass through all of the training data.

Comparison of Batching Approaches



Regularization Techniques for Deep Learning

We have several means by which to help “regularize” neural networks - that is, to prevent overfitting

- Regularization penalty in cost function
    - One option is to explicitly add a penalty to the loss function for having high weights
- Dropout:
    - Dropout is a mechanism where at each training iteration (batch) we randomly remove a subset of neurons

Regularization Techniques for Deep Learning

- Early stopping
  - Another, more heuristical approach to regularization is early stopping.
  - This refers to choosing some rules after which to stop training.
- Stochastic / Mini-batch Gradient descent (to some degree)

Optimizers for training deep learning models

There are several variants to updating the weights which give better performance in practice.

These successive “tweaks” each attempt to improve on the previous idea.

The resulting (often complicated) methods are referred to as **“optimizers”**.

- RMSProp
- Adam

  RMSProp and Adam seem to be quite popular now.

  Difficult to predict in advance which will be best for a particular problem.

Transfer Learning

- Transfer learning is a machine learning method where a model developed for a task is reused as the starting point for a model on a second task.
- Popular approach in deep learning where pre-trained models are used as the starting point on computer vision and natural language processing tasks.
- Several Pre-trained models exist and usually are part of the deep learning libraries e.g Keras.
- These include;
    - VGG, ResNet, Inception, Inception-ResNet

There are two ways to use a pre-trained network:
feature extraction and *fine-tuning*

Caution

Majority of the problems can be solved by classical machine learning algorithms.

Use deep learning only when you really need it.

References

- http://colah.github.io/posts/2015-08-Understanding-LSTMs/
- Deep learning with python book
- Deep learning Specialization (Cousera)
- Deep Learning Book (Ian Goodfellow)
- https://cs231n.github.io/convolutional-networks/
- http://introtodeeplearning.com/slides/6S191_MIT_DeepLearning_L1.pdf

Thank you