Neural Network, Reinforcement Learning and Application

ALAN ZHANG

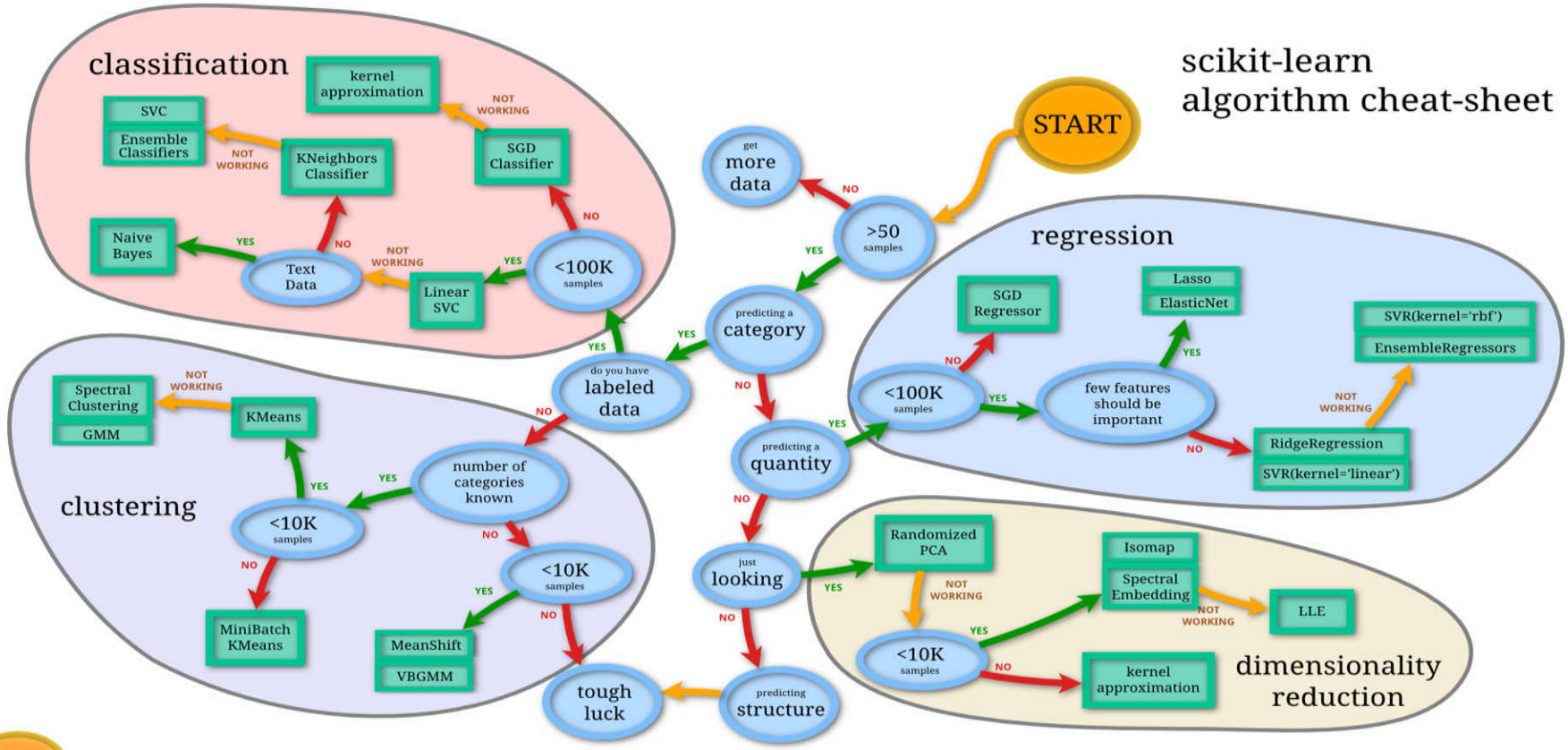
2017/03/23

Agenda

- Basic knowledge on machine learning algo and general application field
- Basic concepts of Tensorflow
- Neural network model and usage in Linear Regression and Logistic Regression
- Demo CNN w/ Tensorflow on MNIST dataset
- Demo RNN w/ Tensorflow on MNIST dataset
- Factors considering in NN usage w/ Google Demo
- Basic knowledge on Reinforcement Learning
- Basic algo in Reinforcement Learning
- 3 x Demo of Reinforcement Learning (2 x Finding Treasure, 1 x armedBandit)

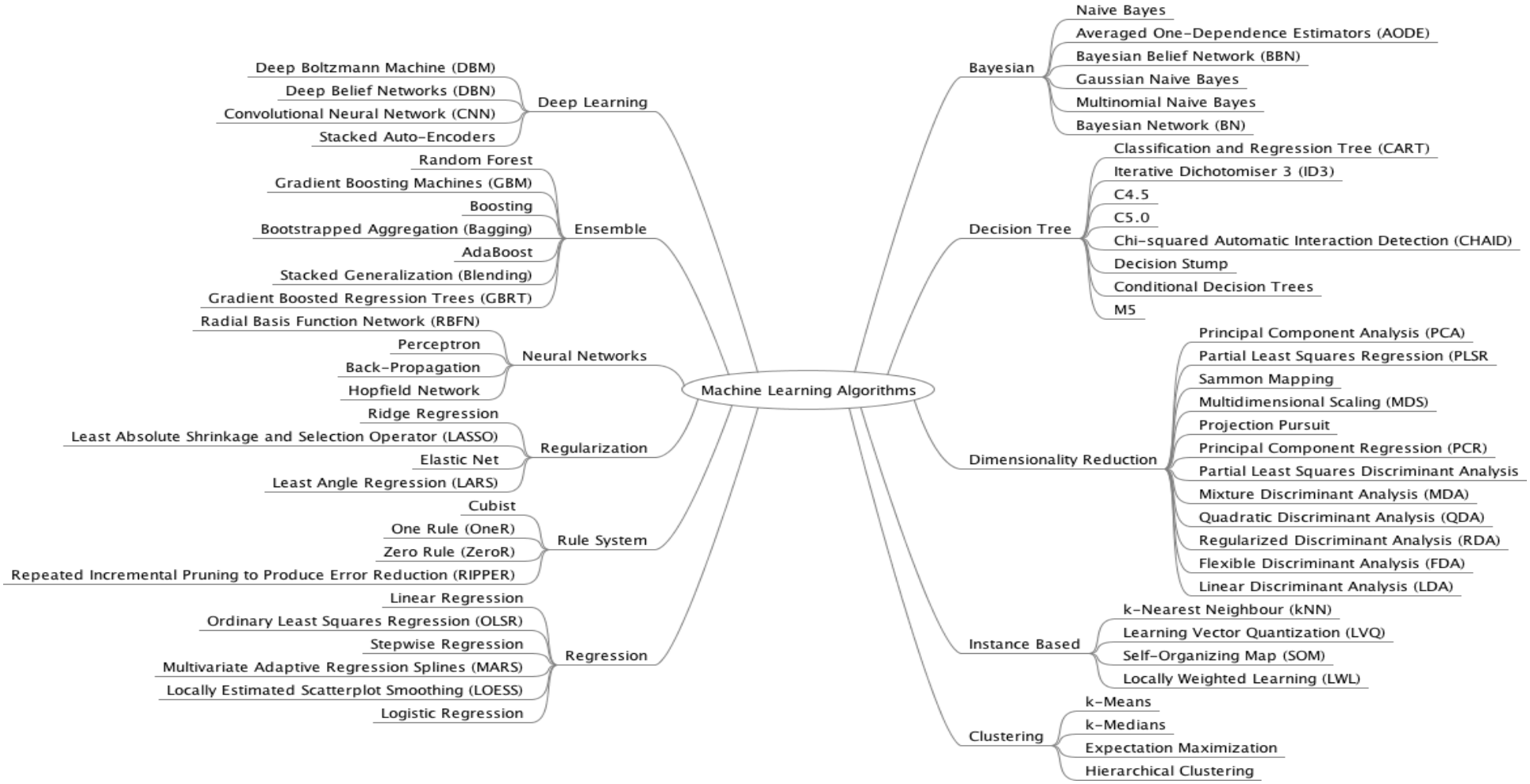
Categories of Machine Learning and Problem Domain

- Supervised Learning
  - Classification and Regression
- Non-supervised Learning
  - Clustering and Reduce Dimensionality
- Reinforcement Learning
  - Autonomous Control, Dynamic Predication etc.



Quote: http://scikit-learn.org/stable/tutorial/machine_learning_map/index.html

Quote: http://machinelearningmastery.com/a-tour-of-machine-learning-algorithms/



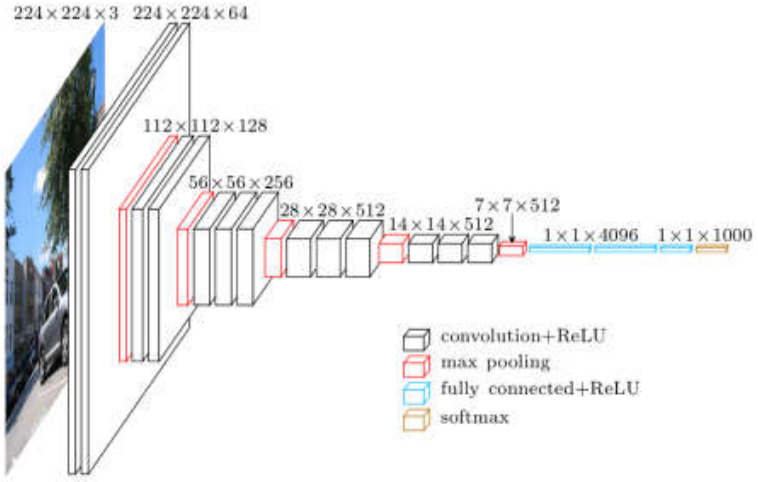
Popular deep learning frameworks

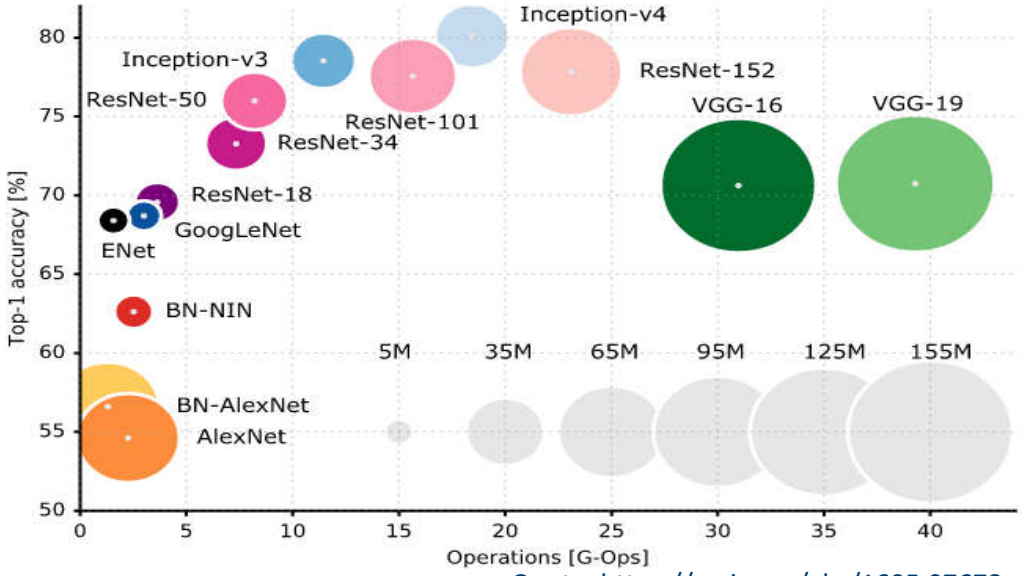
	主语言	从语言	硬件	分布式	命令式	声明式
Caffe	C/C++	Python/Matlab	CPU/GPU	x	x	v
Torch	Lua	C/C++	CPU/GPU/FPGA	x	v	x
Theano	Python	-	CPU/GPU	x	x	v
TensorFlow	Python	C++	CPU/GPU/Mobile	v	x	v
MXNet	C++	Python/R/Julia	CPU/GPU/Mobile	v	v	v

Quote:
https://github.com/zer0n/deepframeworks

Relation between Models and Frameworks

Models like GoogLeNet, AlexNet, VGG are ConvNet architectures constructed on deep learning frameworks such as MXNet, Tensorflow, Torch etc, and their most usage are for Image Recognition, Classification, and compete in ISLVRC, what their differentiations are reflected in performance due to fine tune hardware configuration, software components layers and parameters in the deep learning frameworks.





Quote: https://arxiv.org/abs/1605.07678

Popular ConvNet Architectures

LeNet (1990s): Already covered in this article.

1990s to 2012: In the years from late 1990s to early 2010s convolutional neural network were in incubation. As more and more data and computing power became available, tasks that convolutional neural networks could tackle became more and more interesting.

AlexNet (2012) – In 2012, Alex Krizhevsky (and others) released AlexNet which was a deeper and much wider version of the LeNet and won by a large margin the difficult ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012. It was a significant breakthrough with respect to the previous approaches and the current widespread application of CNNs can be attributed to this work.

ZF Net (2013) – The ILSVRC 2013 winner was a Convolutional Network from Matthew Zeiler and Rob Fergus. It became known as the ZFNet (short for Zeiler & Fergus Net). It was an improvement on AlexNet by tweaking the architecture hyperparameters.

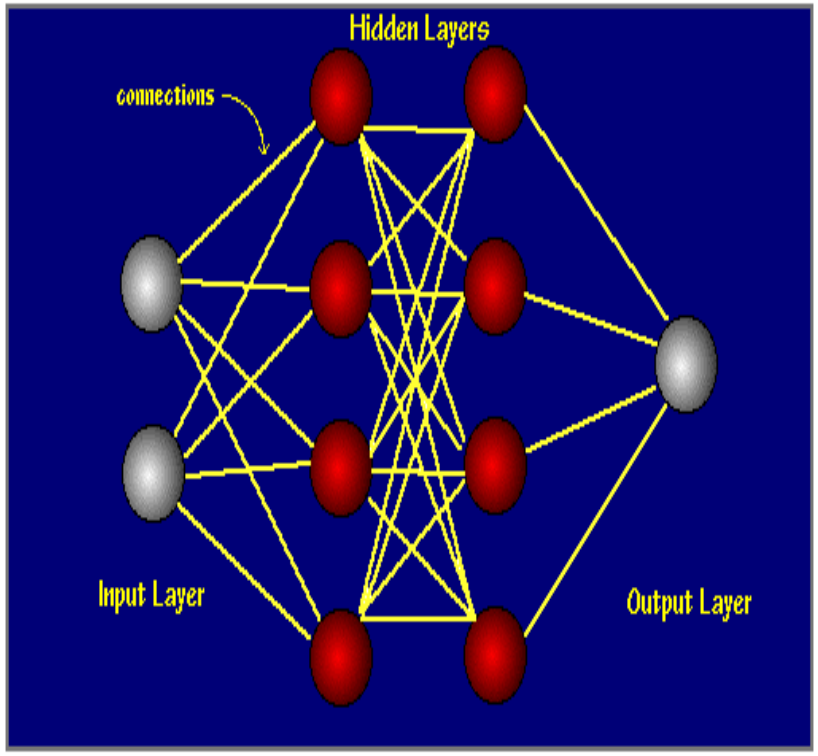
GoogLeNet (2014) – The ILSVRC 2014 winner was a Convolutional Network from Szegedy et al. from Google. Its main contribution was the development of an *Inception Module* that dramatically reduced the number of parameters in the network (4M, compared to AlexNet with 60M).

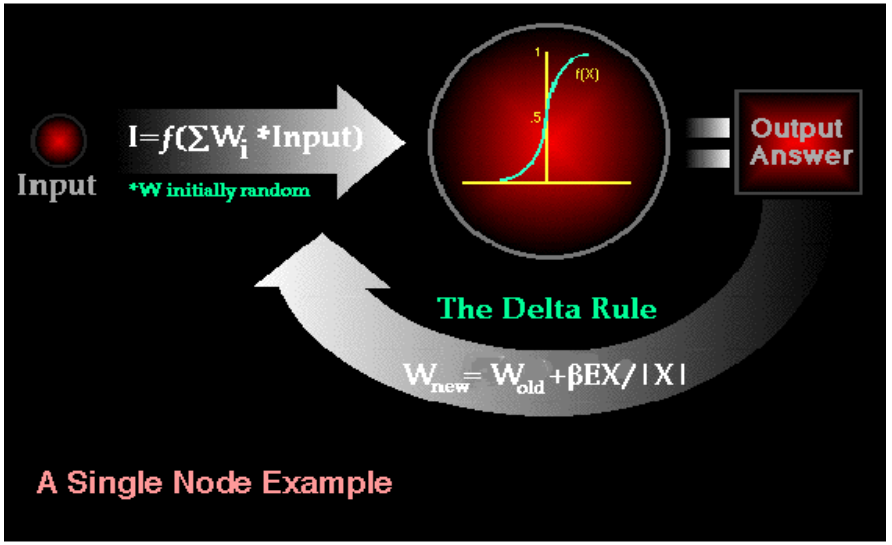
VGGNet (2014) – The runner-up in ILSVRC 2014 was the network that became known as the VGGNet. Its main contribution was in showing that the depth of the network (number of layers) is a critical component for good performance.

ResNets (2015) – Residual Network developed by Kaiming He (and others) was the winner of ILSVRC 2015. ResNets are currently by far state of the art Convolutional Neural Network models and are the default choice for using ConvNets in practice (as of May 2016).

DenseNet (August 2016) – Recently published by Gao Huang (and others), the Densely Connected Convolutional Network has each layer directly connected to every other layer in a feed-forward fashion. The DenseNet has been shown to obtain significant improvements over previous state-of-the-art architectures on five highly competitive object recognition benchmark tasks. Check out the Torch implementation here.

Neural Network and Neuron





Three key elements consisting of neuron: Weight, Bias, Activation Function, whereas Weight and Bias can be automatically tuned during training process via backpropagation algorithm;
Activation Function: Softmax, Sigmoid, Tanh, and ReLU have been experienced in specific field applications. Softmax is used in output layer to convert one K-dimensional vector of any real values to anther K-dimensional vector of real value range (0, 1).

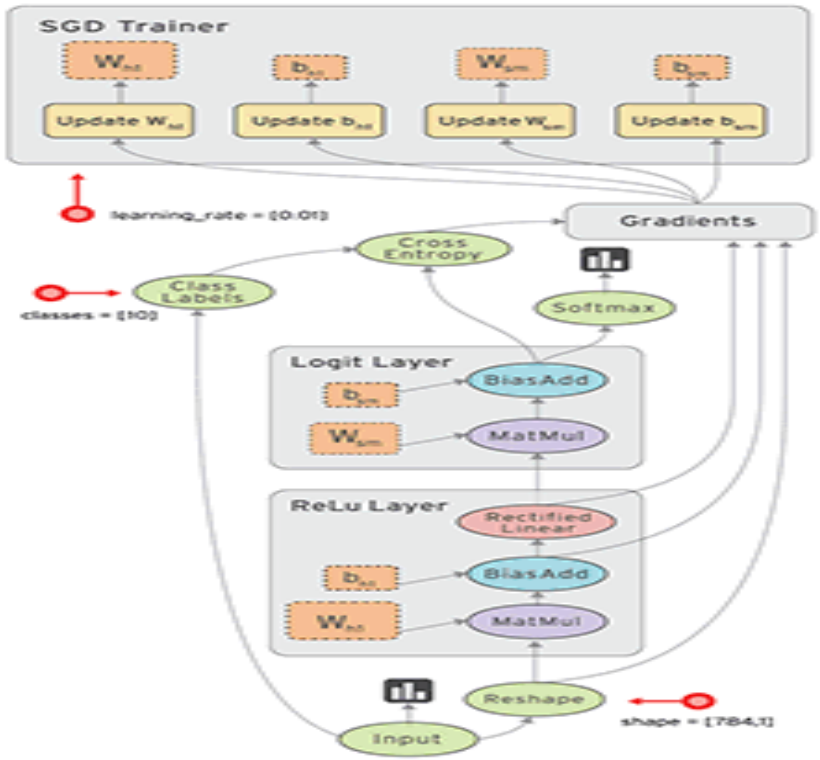
General method of using neural network

- Extract the major features of training data (data source can be divided into training dataset and test dataset) via Autoencoder and Clustering processing, Both Autoencoder and Clustering is Non-surprised learning process in Dimensionality Reduction, and highlight Density Distribution, and normalize the input format to the input layer of the designated neural network;
- Set up layers of the designated neural network, and assign random Weights & Biases for each nodes inside every layers, and Activation function such as Sigmoid, ReLU, Tanh etc.;
- Define Loss function such as Square Loss for Linear Regression, Cross-entropy Loss for Logic Regression (i.e. Classifier), Hinge Loss for Support Vector Machine (SVM) etc.;
- Set up training model of the designated neural network, i.e. specify Optimizer function such as SGD, Momentum, NAG, Adagrad, Adadelta, Rmsprop etc; (Quota:https://www.tensorflow.org/api_docs/python/tf/train)
- Enter training loop and output Loss value during epochs, then run verification with test dataset, output the accuracy based on trained parameters.

Basic knowledge on Tensorflow

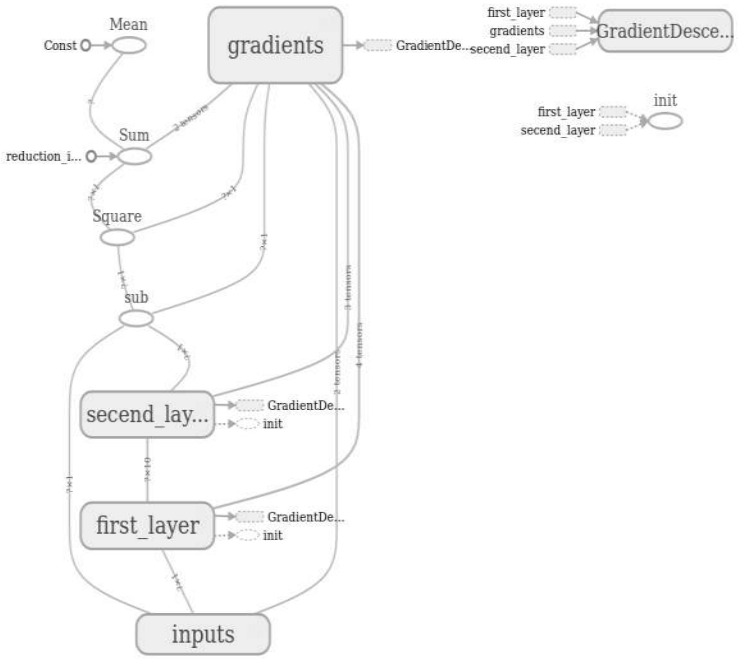
Key concepts in Tensorflow

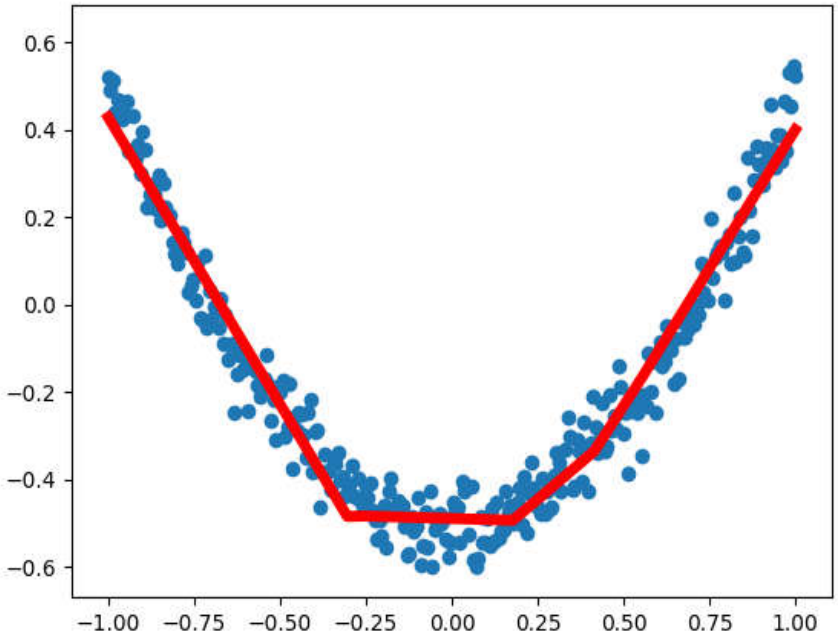
- TensorFlow: is a programming system in which you represent computations as graphs in which the nodes represents math operations.
- Tensor: an n-dimensional array or list. Only tensors may be passed between nodes in the computation graph.
- Session: Sessions contain "computational graphs" that represent calculations to be carried out.
- Placeholder: An interface between a computational graph element and your data. Placeholder can present input or output.
- The defined "Graph Model" will be processed by the TensorFlow engine when Session.run() is called.
- TensorFlow can support graphic view of computing structure that can help to deploy computations to several CPUs/GPUs.



https://www.tensorflow.org/images/tensors_flowing.gif

Using Tensorflow on Linear Regression





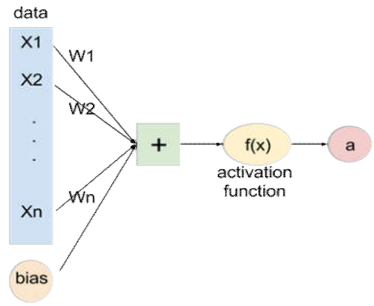
test4tf_demoTensorBoard.py

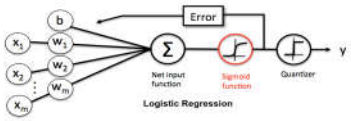
Using Tensorflow on Logistic Regression

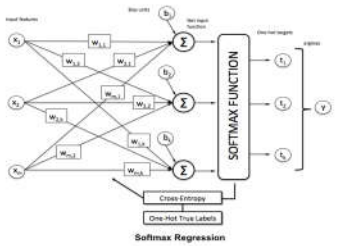
Essentially, neuron math model is identical to logistic regression math model; multi-layers neural network is a large scale and complicated logistic regression network.

Mathematical Model：$s = \sum_{n=1}^{\infty}(\theta_n x_n)$
Activation Function: Sigmoid Function (Logistic Function): Map the summary s of weighted inputs to be a float number between 0 ~ 1 to reflect the possibility of certain class.
Loss Function: Log Loss Function, i.e. Cross-Entropy Loss, quantify the gap between real result and predicted result.

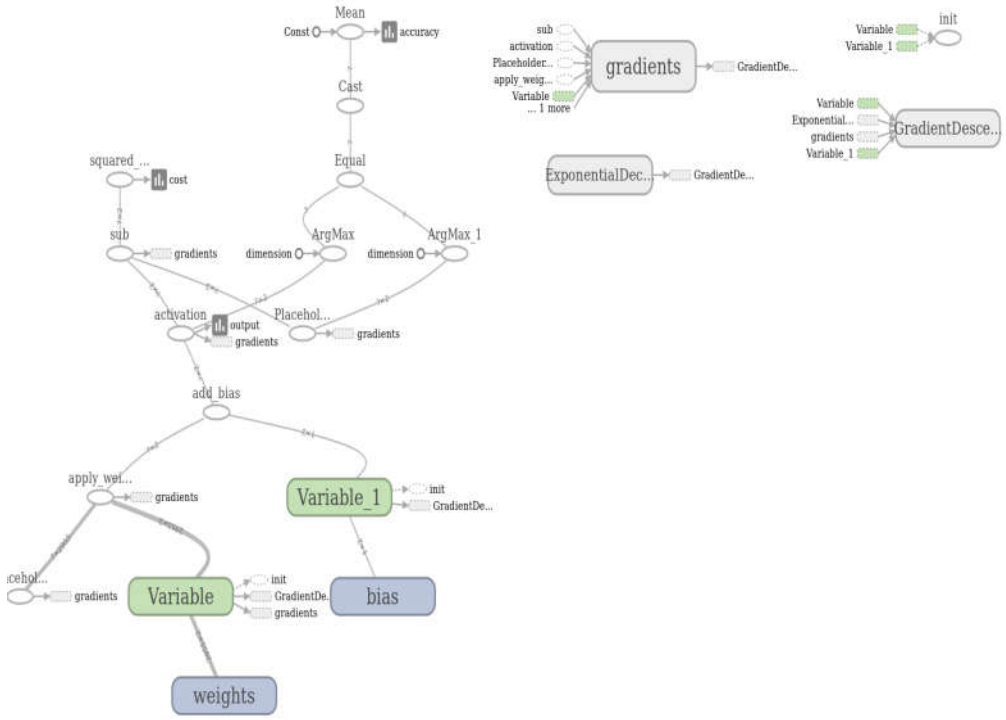
交叉熵=事件的真实分布+不可预测性，所以交叉熵可以用于度量两个概率分布（真实分布&预测分布）之间的差异性，即：交叉熵损失函数（对数损失函数）可以衡量一个模型对真实值带来的额外噪音，通过最小化交叉熵损失函数（对数损失函数），我们就可以最大化分类器（模型）的精度。用Cross-Entropy Loss的数学依据是：Cross-Entropy Loss是关于参数向量θ的凸函数，利于求局部最优解。
Solution Algorithm: 使用Stochastic gradient descent (SGD)，Limited-memory BFGS (L-BFGS) 等实现最优参数θ的求解。



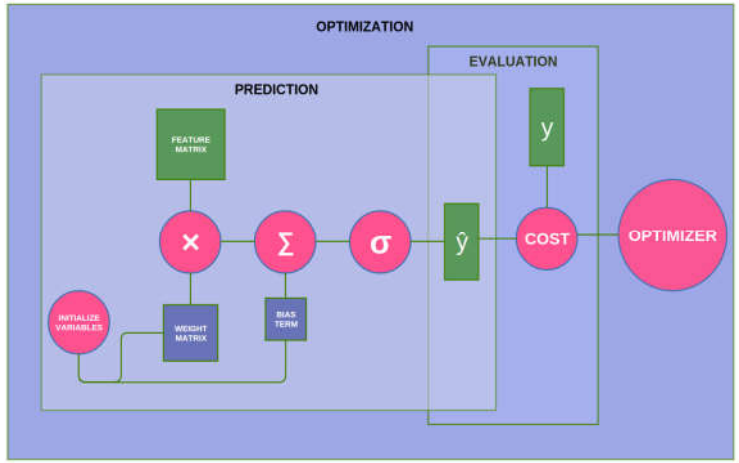




Using Tensorflow on Logistic Regression



Example: Email classification
The left diagram shows the computing graph of logistic_regression_train.py and logistic_regression_predict.py.



Quote: http://jrmeyer.github.io/tutorial/2016/02/01/TensorFlow-Tutorial.html

CNN for MNIST

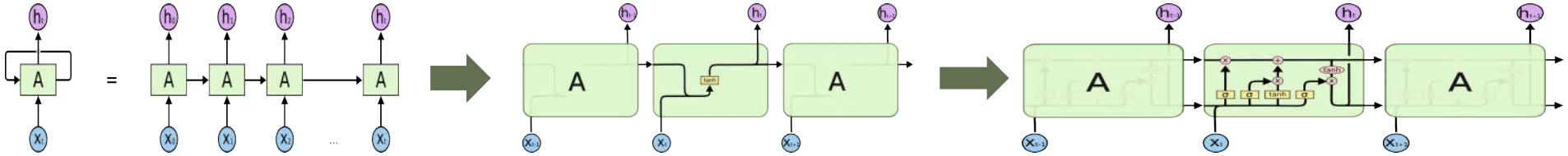
Convolutional neural networks (CNNs) are the current state-of-the-art model architecture for image classification tasks. CNN contains 3 components:

- ❑ Convolution Layers, which apply a specified number of convolution filters (alias name: Filter, Kernel, Feature Detector) such as “Identity”, “Edge Detect”, “Sharpen”, “Box Blur”, “Gaussian Blur” to the image, to produce a single value in the output feature map (alias name: Convolve Feature, Activation Map). Convolutional layers then typically apply a ReLU activation function to the output to introduce nonlinearities into the model.
- ❑ Pooling Layer, which downsample the image data extracted by the convolutional layers to reduce the dimensionality of the feature map in order to decrease processing time.
- ❑ Dense (Fully connected) Layer, which perform classification on the features extracted by the convolutional layers and downsampled by the pooling layers.

(n^{th}_sample, 28 x 28, 1 channel) → [5 x 5 pitch (conv filter/kernel)]$_{32\ neuron}$ → [2 x 2 Max Pooling]$_{32\ neuron}$ → (n^{th}_sample, 14 x 14, 1 channel)$_{32\ neuron}$→ [5 x 5 pitch (conv filter/kernel)]$_{64\ neuron}$ → 2 x 2 Max Pooling → (n^{th}_sample, 7 x 7, 1 channel)64 neuron → (FCL_1 output 1024 1-D data)$_{7\ x\ 7\ x\ 64\ neuron}$ → (FCL_2 output 10 1-D data)$_{1024\ neuron}$ → Softmax output

Example: test4tf_cnn.py
Quote: https://www.tensorflow.org/versions/master/tutorials/layers/
http://robromijnders.github.io/tensorflow_basic/
https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

RNN for MNIST



LSTM RNN Module: Gating, Mux, Update, Output Decision, 4 neural network layers.

Demo: test4tf_rnn.py

Reference and Quote:
http://karpathy.github.io/2015/05/21/rnn-effectiveness/
http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Factors considering in NN implementation

For neuron:

- ❑ Activation Function: Sigmoid, Tanh, ReLU, Softmax (for output);
- ❑ Weight & Bias: adjusted automatically during training process.

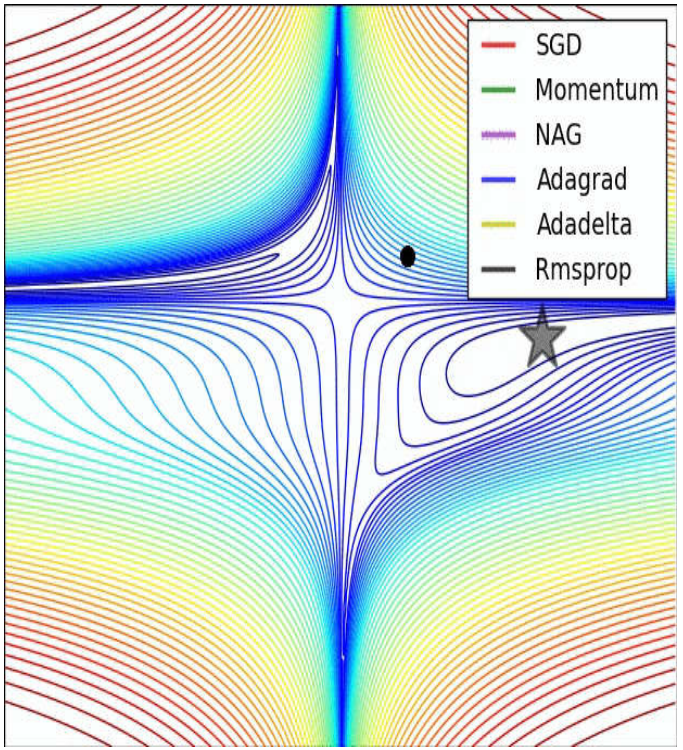
For NN:

- ❑ Over-fitting: Adjust neuron output drop-out probability to simulate real human neuron behavior;
- ❑ Learning rate: Referenced by Optimizer algo that instruct how fast the weight of neuron can be evolved. Setting high may miss the most optimized result, setting low may cause optimizer take long time to search optimized result. A general practical guide to decay Learning Rate by 0.5 ratio every 30~50 epochs.
- ❑ ConvFilter: Pitch size, Stride, Max/Average Pool
- ❑ Optimizer algo: Refer to https://www.tensorflow.org/api_guides/python/train to see what optimizer algo can support

For data samples:

- ❑ Normalization: If the input dataset span in big range or small range that may cause activation function output enter saturating area, please refer to https://www.tensorflow.org/versions/r0.12/api_docs/python/nn/normalization to understand what normalization algo can support.

Quote: Animation credit: Alec Radford
Demo: A Neural Network Playground



Key Concept in Reinforcement Learning

Differing from other machine learning algorithms that are used to process input data set, and draw out major features for classification, regression, clustering, and reduce dimensionality, Reinforcement Learning is used to perform autonomous control and dynamic forecast in return way to outside world (i.e. environment).

Roles in Reinforcement Learning

- Agent: the actor;
- Environment: the set of system state.

Major Character of Reinforcement Learning

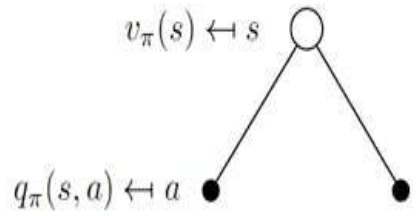
- "Trail-and-Error" to find out the optimized Policies to bring about best/optimized accumulated Reward. Policy is the map between State and Action, generally, it should be a possibility of certain action 'a' under certain state 's';
- Delay Reward, the future reward is estimated by stochastic evaluation of future executing policies described in Action-Value function (i.e. Table);
- Discrete time, discrete observations, and discrete actions.

Training Method

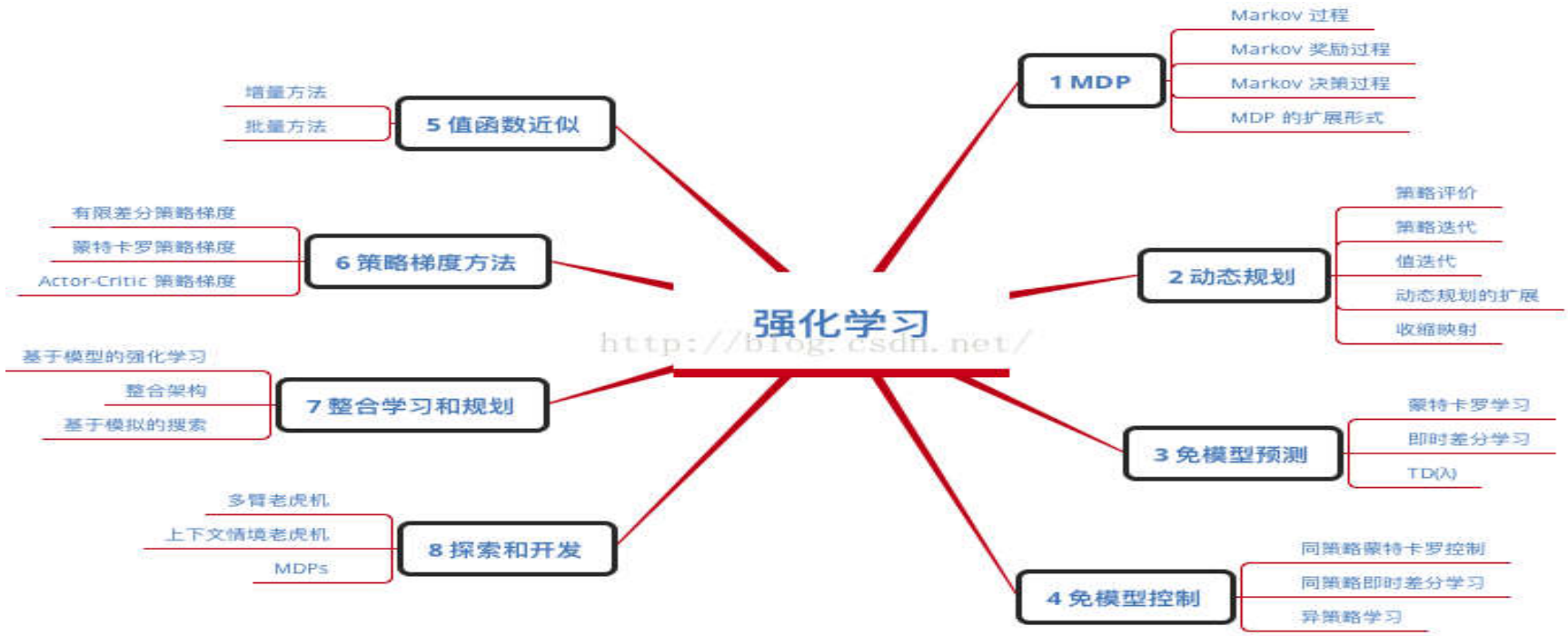
- Iterate Policy and State with State-Reward to evolve State-Value function, and Action-State Value function to get max value of Action-State Value function under any initial state;
- State-Reward defines the rule of game, i.e. system of reward and penalty; Q-Table (the result of evolved State-Value function, and Action-State Value function) is the intelligence of Agent.

Relation between State-Value function Action-State Value function

$$q_\pi(s,a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a \sum_{a' \in A} \pi(a'|s') q_\pi(s',a') = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_\pi(s')$$



Reinforcement Learning Algorithm Map



Quote: http://blog.csdn.net/ikerpeng/article/details/53031551

Most Used Algorithms in Reinforcement Learning

- Dynamic Programming method
  - Fully model-based (reward function, state transition probabilities, and fixed policy), execute policy iteration in policy evaluation and improvement;
  - Traverse for full possibilities to get State-Value and Q-Function, update per step (online mode).

- Monte Carlo method
  - Model-free, learning from samples and calculate an approximate optimized Q-Function for a specific state independently, episode-based (batch mode).

- Temporal Difference method
  - Step-based update (online mode), model-free.
  - Q-Learning (Demo: test4tf_qLearning.py, test4qLearningRun.py, test4n_armedBanditProblem.py)
  - Sarsa

Reference & Quote Resources:
https://github.com/stone8oy/deepRL/tree/resource
Reinforcement Learning-An Introduction
Algorithms for Reinforcement Learning
Reinforcement Learning State-of-the-Art
Recent Advances in Reinforcement Learning