Guide to training neural networks (Mostly working tricks)

(Notes from lectures of Geff Hinton and Andrew Ng)

==========

1. Data pre-processing
    a. mean and variance normalization
        i. Have data in range (-1,1) which makes network operate in linear region and network being linear won't overfit. Learning becomes faster with tanh activation
    b. Variance scaling makes error surface circular and speeds up training
    c. Decorrelation of input
        i. Pca - by dropping smallest eigenvalues and divide remaining by square root of their eigenvalues, this gives a circular error surface and makes learning faster incase of linear activation
2. Start with bare minimum network
3. Network initialization
    a. Weight initialization - used to make learning easier and break the symmetry of neurons to learn different features.
        i. Use 'xavior/glorot_uniform' if activation function is tanh or sigmoid
        ii. Use 'he_uniform' if activation function is Relu
4. Choose learning rate that decays smoothly over training loss and reasonably on validation loss
    a. Prefer Adam as optimization algorithm as it covers both momentum and RMSprop, helps weight updates in the right direction and can work with higher learning rates
    b. Adadelta is a generalization of Adam, it helps and doesn't depend on learning rate
    c. Tune learning rate
        i. Training loss is oscillating heavily then reduce learning rate
        ii. If Training loss decreasing slowly then there can be fluctuations on validation loss - give it some time to check if validation loss fluctuations are settling and training decreasing slowly and consistently.
        iii. If training loss decreasing slowly and consistently then increase learning rate
        iv. If validation loss stops decreasing reduce learning rate
        v. Starting with higher learning rate will cause weights of hidden units to be large positive or large negative which will make neurons to saturate and gradients will vanish and network stop learning (train/validation error will not decrease) and network is stuck at the plateau (this is true in case of tanh/sigmoid). (NOTE - recent studies shows that getting stuck in a local minima is

very very rare so when network stops learning it's most likely a plateau).

vi. If you start learning quickly and at a later point training/validation error stops decreasing then it’s an indication that network is stuck in a plateau

vii. If error is not decreasing then decreasing LR will help but if LR is decreased too soon we may get a quick win but later it will stop decreasing error and network will get stuck in a plateau.

5. Overfit the network
    a. Increase network capacity (neurons and layers)
6. Regularize the network

    Overfitting is mainly due to two issues, and it will occur when the network start learning these details (this is assumed to be random):

    1. Labeling error - where out of the training sample is wrong
    2. Sampling error - Input i.e. sample data is wrong

    a. Collect more data
    b. Data augmentation
        i. For images flipping, cropping, shearing, distortion/noise etc.,
    c. Reduce network capacity (hidden layers and number of neurons) to make it learn important features
    d. Early stopping by starting with small weights and stop before network overfits (It might help to restore best weights and start again ***)
        i. Small weights doesn’t have much capacity (network will be linear)
    e. L2 norm (not very effective in NN) will cause weight decay and reduces network capacity (Also similar to adding noise to input)
    f. L1 norm (capacity reduction)
    g. Dropout
        i. Multiplicative noise acts as regularizer
    h. Batch normalization
        i. Normalize hidden units inputs which will speed up learning
        ii. Make next layer hidden units invariant of weight changes of previous layers, helps generalize better
        iii. Mean and variance noise (added due to scaling) acts as regularizer