Improving the way neural networks learn

Subjects

- Cost function
- Layer activation function
- Regularization
- Activation functions
- Weight initialization
- How to choose the hyper-parameters
- Other techniques

Cost function

Cross-entropy cost function

$C = -\frac{1}{n}\sum_x \sum_j \left[y_j \ln a_j^L + (1 - y_j) \ln(1 - a_j^L) \right]$

Use

Combine with sigmoid to avoid slow learning at high cost

Where does it come from?

- Integrate from derivative cost function without sigmoid
- Measure of surprise for using a, but expecting y

Layer activation function

Softmax

$$a_j^L = \frac{e^{z_j^L}}{\sum_k e^{z_k^L}}$$

Uses

- Gives a probability distribution
- Combines well with log likelyhood $C \equiv -\ln a_y^L$

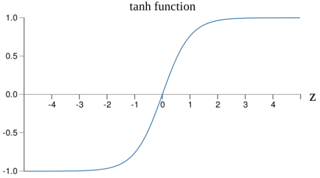
Regularization

Regularization

To prevent overfitting, add term to cost function

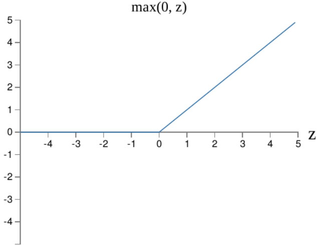
- L1: $\frac{\lambda}{n}\sum_w |w|$
- L2: $\frac{\lambda}{2n}\sum_w w^2$

Activation functions



Activation functions

- tanh
- rectified linear



Weight initialization

Weight initialization

The old distribution gave a very broad expectation for z in the next layer. Initialize them with $1/\sqrt{n_{\mathrm{in}}}$, and the resulting distribution of z will have a standard deviation of 1.22.

Choose good hyper-parameters

Choosing good hyper-parameters

- Can be really difficult if there is no learning at all
- Simplify the problem to get some result
- Simplify the network, add monitoring
- Choose learning rate between oscillation and slow learning
- Early stopping for epochs - no improvement in ten rule

Other techniques

Other techniques

- Hessian
- Momentum based
- Dropout
- Artificial expansion of training data
- Variable learning rate