Training Neural Networks

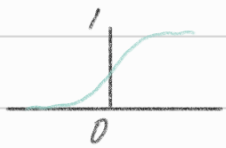
Mini-batch SGD

Loop:
1. Sample a batch of data
2. Forward prop it through the graph, get loss
3. Backprop to calculate the gradients
4. Update the parameters using the gradients

Activation Functions

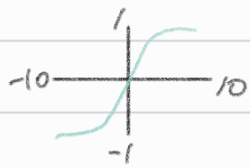
Sigmoid

$\sigma(x) = \frac{1}{(1+e^{-x})}$



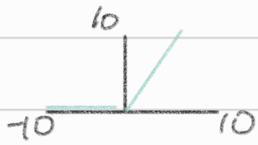
- saturates "firing rate" of a neuron
- Saturated neurons kill the gradients
- Sigmoid outputs are not zero-centered
- exp() is a bit compute expensive

tanh(x)



- squashes numbers to [-1, 1]
- zero centered
- still kills gradients when saturated

ReLU



- Computes $f(x) = \max(0, x)$
- does not saturate
- very computationally efficient
- Converges much faster than sigmoid, tanh
- actually more biologically plausible than sigmoid.
- Not zero-centered

Maxout "Neuron" $\max(w_1^T x + b_1, w_2^T x + b_2)$

- not the form of dot product : nonlinearity
- generalizes ReLU and leaky ReLU
- Linear regime, does not saturate/die
- doubles the number of parameters/neuron

결론: 실전에서 ReLU가 가장 좋은 성능

Data Preprocessing

in practice for images : center only

not common to normalize variance, to do PCA or whitening

Weight Initialization

1st idea : Small random numbers

(가우시안 분포 0평균 1e-2 표준편차)

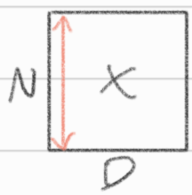
Deep layer → all activations become zero

"Xavier Initialization" "He"

Batch Normalization

(for unit gaussian activations) Variance를 줄이기위해

1. compute the empirical mean and variance independently for each dimension



2. Normalize

$$\hat{x}^{(k)} = \frac{x^{(k)} - E[x^{(k)}]}{\sqrt{Var[x^{(k)}]}}$$

∴ Improves gradient flow through the network
Allows higher learning rates
Reduces the strong dependence on initialization
Regularization Effect

Z-score Normalization : (X - 평균) / 표준편차

Random Search > Grid Search

Loss curve visualization 으로 learning rate 조정