Machine Learning: Yoshinari Fujinuma

University of Colorado Boulder

LECTURE 16

Slides adapted from Chenhao Tan, Jordan Boyd-Graber, Justin Johnson, Andrej Karpathy, Chris Ketelsen, Fei-Fei Li, Mike Mozer, Michael Nielson

Logistics

- Releasing the guideline for the final proposal on Friday (Proposal will be due on 3/19)

Overview

Back propagation recap with an Example

Practical issues of back propagation

- Vanishing Gradients
- Weight Initialization
- Alternative regularization (Dropout)
- Batch size

Outline

Back propagation recap with an Example

Practical issues of back propagation

- Vanishing Gradients
- Weight Initialization
- Alternative regularization (Dropout)
- Batch size

Back Propagation Summary

$\delta^L = \frac{\partial \mathscr{L}}{\partial a_j^L} \odot g'(\mathbf{z}^L)$ # Compute δ's on output layer

For $\ell = L, \ldots, 1$

$\frac{\partial \mathscr{L}}{\partial \mathbf{W}^\ell} = \boldsymbol{\delta}^\ell (\boldsymbol{a}^{l-1})^T$ # Compute weight derivatives

$\frac{\partial \mathscr{L}}{\partial \boldsymbol{b}^\ell} = \boldsymbol{\delta}^\ell$ # Compute bias derivatives

$\boldsymbol{\delta}^{\ell-1} = \left(W^\ell\right)^T \boldsymbol{\delta}^\ell \odot g'(\mathbf{z}^{\ell-1})$ # Back prop $\boldsymbol{\delta}$'s to previous layer

Let's recap where does this $\boldsymbol{\delta}$ came from...

Back Propgation Example with two hidden layers, one unit

We define δ as the derivative w.r.t. the z's by δ:

$$\delta_j^l = \frac{\partial \mathscr{L}}{\partial z_j^l}$$

Let's assume we use sigmoid function σ for the activation function

$$a^0 = x$$

$$z^1 = w^1 a^0 + b^1$$
$$a^1 = \sigma(z^1)$$

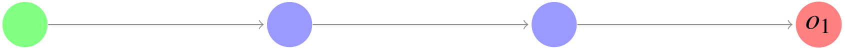
$$z^2 = w^2 a^1 + b^2$$
$$a^2 = \sigma(z^2)$$

$$\hat{y} = a^2$$

Back Propgation Example with two hidden layers, one unit

Forward Propagation

$$a^0 = x \qquad \begin{aligned} z^1 &= w^1 a^0 + b^1 \\ a^1 &= \sigma(z^1) \end{aligned} \qquad \begin{aligned} z^2 &= w^2 a^1 + b^2 \\ a^2 &= \sigma(z^2) \end{aligned} \qquad \hat{y} = a^2$$



Back Propgation Example with two hidden layers, one unit

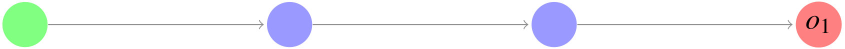
Forward Propagation

$$a^0 = x$$

$$z^1 = w^1 a^0 + b^1$$
$$a^1 = \sigma(z^1)$$

$$z^2 = w^2 a^1 + b^2$$
$$a^2 = \sigma(z^2)$$

$$\hat{y} = a^2$$



and further assume we use squared loss function L

Back Propgation Example with two hidden layers, one unit

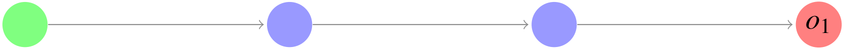
Forward Propagation

$$a^0 = x$$

$$z^1 = w^1 a^0 + b^1 \qquad a^1 = \sigma(z^1)$$

$$z^2 = w^2 a^1 + b^2 \qquad a^2 = \sigma(z^2)$$

$$\hat{y} = a^2$$



and further assume we use squared loss function L

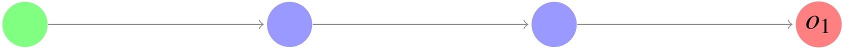
Back Propagation

$$a^0 = x$$

$$\frac{\partial z^1}{\partial w^1} = a^0 \qquad \frac{\partial a^1}{\partial z^1} = \sigma(z^1)(1-\sigma(z^1))$$

$$\frac{\partial z^2}{\partial w^2} = a^1 \qquad \frac{\partial a^2}{\partial z^2} = \sigma(z^2)(1-\sigma(z^2))$$

$$L = (a^2 - y)^2 \qquad \frac{\partial L}{\partial a^2} = 2(a^2 - y)$$



Back propgation with two hidden layers, one unit

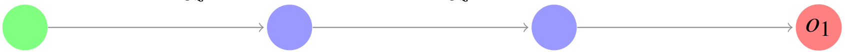
Back Propagation

$$a^0 = x$$

$$\frac{\partial z^1}{\partial w^1} = a^0 \qquad \frac{\partial a^1}{\partial z^1} = \sigma(z^1)(1-\sigma(z^1))$$

$$\frac{\partial z^2}{\partial w^2} = a^1 \qquad \frac{\partial a^2}{\partial z^2} = \sigma(z^2)(1-\sigma(z^2))$$

$$L = (a^2 - y)^2 \qquad \frac{\partial L}{\partial a^2} = 2(a^2-y)$$



E.g., we need $\frac{\partial L}{\partial w^1}$ and $\frac{\partial L}{\partial w^2}$ to update the parameters w^1 and w^2 using SGD, so

- $\frac{\partial L}{\partial w^1} = \frac{\partial z^1}{\partial w^1} \underbrace{\frac{\partial a^1}{\partial z^1} \frac{\partial z^2}{\partial a^1} \frac{\partial a^2}{\partial z^2} \frac{\partial L}{\partial a^2}}_{\delta^1} = a^0 \delta^1$

Back propgation with two hidden layers, one unit

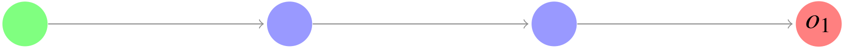
Back Propagation

$$a^0 = x$$

$$\frac{\partial z^1}{\partial w^1} = a^0 \qquad \frac{\partial a^1}{\partial z^1} = \sigma(z^1)(1-\sigma(z^1))$$

$$\frac{\partial z^2}{\partial w^2} = a^1 \qquad \frac{\partial a^2}{\partial z^2} = \sigma(z^2)(1-\sigma(z^2))$$

$$L = (a^2 - y)^2 \qquad \frac{\partial L}{\partial a^2} = 2(a^2-y)$$



E.g., we need $\frac{\partial L}{\partial w^1}$ and $\frac{\partial L}{\partial w^2}$ to update the parameters w^1 and w^2 using SGD, so

- $\frac{\partial L}{\partial w^1} = \frac{\partial z^1}{\partial w^1} \underbrace{\frac{\partial a^1}{\partial z^1} \frac{\partial z^2}{\partial a^1} \frac{\partial a^2}{\partial z^2} \frac{\partial L}{\partial a^2}}_{\delta^1} = a^0 \delta^1$
- $\frac{\partial L}{\partial w^2} = \frac{\partial z^2}{\partial w^2} \underbrace{\frac{\partial a^2}{\partial z^2} \frac{\partial L}{\partial a^2}}_{\delta^2} = a^1 \delta^2$

Outline

Back propagation recap with an Example

Practical issues of back propagation

- Vanishing Gradients
- Weight Initialization
- Alternative regularization (Dropout)
- Batch size

Back Propagation

In practice, many remaining questions may arise.

$\delta^L = \frac{\partial \mathscr{L}}{\partial a_j^L} \odot g'(\mathbf{z}^L)$ # Compute δ's on output layer

For $\ell = L, \ldots, 1$

$\frac{\partial \mathscr{L}}{\partial \boldsymbol{W}^\ell} = \boldsymbol{\delta}^\ell (\boldsymbol{a}^{l-1})^T$ # Compute weight derivatives

$\frac{\partial \mathscr{L}}{\partial \boldsymbol{b}^\ell} = \boldsymbol{\delta}^\ell$ # Compute bias derivatives

$\boldsymbol{\delta}^{\ell-1} = \left(W^\ell\right)^T \boldsymbol{\delta}^\ell \odot g'(\mathbf{z}^{\ell-1})$ # Back prop $\boldsymbol{\delta}$'s to previous layer

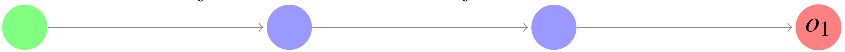
An Example of using Sigmoid Function

$$a^0 = x$$

$$\frac{\partial z^1}{\partial w^1} = a^0 \qquad \frac{\partial a^1}{\partial z^1} = \sigma(z^1)(1-\sigma(z^1))$$

$$\frac{\partial z^2}{\partial w^2} = a^1 \qquad \frac{\partial a^2}{\partial z^2} = \sigma(z^2)(1-\sigma(z^2))$$

$$L = (a^2 - y)^2 \qquad \frac{\partial L}{\partial a^2} = 2(a^2-y)$$



E.g., you need $\frac{\partial L}{\partial w^1}$ to update the parameters w^1 using SGD.

Assume $x = 3$, $w^1 = 1$, $b^1 = -2$, $w^2 = 0.5$, $b^2 = 0.7$.

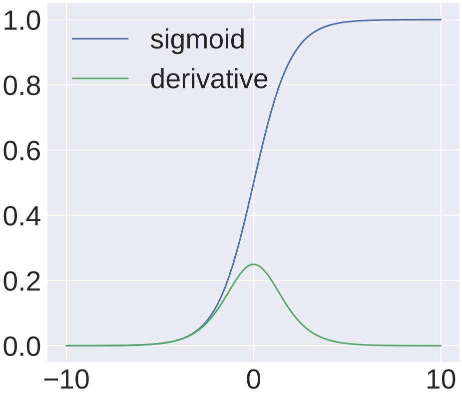
$a^0 = x = 3$,

$z^1 = 1 \cdot 3 - 2 = 1$,

$\frac{\partial a^1}{\partial z^1} = \sigma(z^1)(1 - \sigma(z^1)) = 0.731 \cdot 0.269 < \frac{1}{4}$(and so as $\frac{\partial \mathbf{a}^2}{\partial \mathbf{z}^2}$)

- $\frac{\partial L}{\partial w^1} = \frac{\partial z^1}{\partial w^1} \underbrace{\frac{\partial a^1}{\partial z^1} \frac{\partial z^2}{\partial a^1} \frac{\partial a^2}{\partial z^2} \frac{\partial L}{\partial a^2}}_{\delta^1} = a^0 \delta^1$

Vanishing gradients



Vanishing gradients

If we use Gaussian initialization for weights, $w^j \sim \mathcal{N}(0, 1)$,

$$|w^j| < 1$$

$$|w^j \sigma'(z_j)| < \frac{1}{4}$$

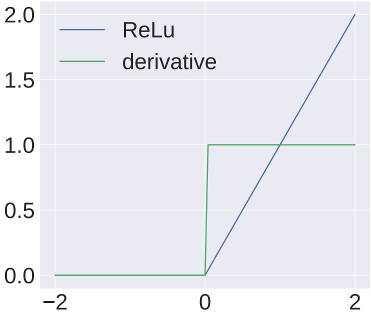
$\frac{\partial \mathcal{L}}{\partial b^1}$ decay to zero exponentially as we add more layers with sigmoid activation function

Assuming $a^0 < 1$,

$\frac{\partial \mathcal{L}}{\partial w^1}$ decay to zero exponentially as we add more layers with sigmoid activation function

Vanishing gradients

ReLU
Gradient is either 0 or 1

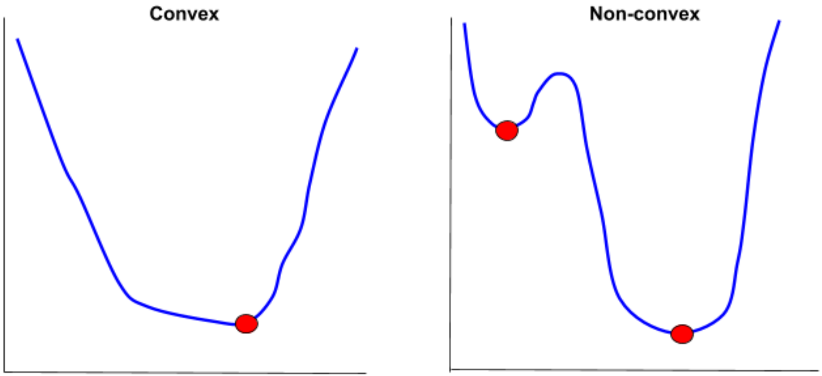


Training a Feed-Forward Neural Network

In practice, many remaining questions may arise, more examples:

1. How do we initialize weights and biases?
2. How do we regularize?
3. Can I batch this?

Non-convexity



Weight initialization

Old idea: $W = 0$, what happens?

Weight initialization

Old idea: $W = 0$, what happens?
There is no source of asymmetry. (Every neuron looks the same and leads to a slow start.)

Weight initialization

Old idea: $W = 0$, what happens?

There is no source of asymmetry. (Every neuron looks the same and leads to a slow start.)

$\delta^L = \nabla_{\boldsymbol{a}^L} \mathscr{L} \odot g'(\mathbf{z}^L)$ # Compute δ's on output layer

For $\ell = L, \ldots, 1$

$\frac{\partial \mathscr{L}}{\partial \mathbf{W}^\ell} = \boldsymbol{\delta}^\ell (\boldsymbol{a}^{l-1})^T$ # Compute weight derivatives

$\frac{\partial \mathscr{L}}{\partial \boldsymbol{b}^\ell} = \boldsymbol{\delta}^\ell$ # Compute bias derivatives

$\boldsymbol{\delta}^{\ell-1} = \left(W^\ell\right)^T \boldsymbol{\delta}^\ell \odot g'(\mathbf{z}^{\ell-1})$ # Back prop $\boldsymbol{\delta}$'s to previous layer

Weight initialization

Small random numbers, $W \sim \mathcal{N}(0, 0.01)$

Weight initialization

Small random numbers, $W \sim \mathcal{N}(0, 0.01)$

Xavier initialization [Glorot and Bengio, 2010]

$$W \sim \mathcal{N}(0, \frac{2}{n_{in} + n_{out}})$$

Weight initialization

Small random numbers, $W \sim \mathcal{N}(0, 0.01)$

Xavier initialization [Glorot and Bengio, 2010]

$$W \sim \mathcal{N}(0, \frac{2}{n_{in} + n_{out}})$$

He initialization [He et al., 2015]

$$W \sim \mathcal{N}(0, \frac{2}{n_{in}})$$

Weight initialization

Small random numbers, $W \sim \mathcal{N}(0, 0.01)$

Xavier initialization [Glorot and Bengio, 2010]

$$W \sim \mathcal{N}(0, \frac{2}{n_{in} + n_{out}})$$

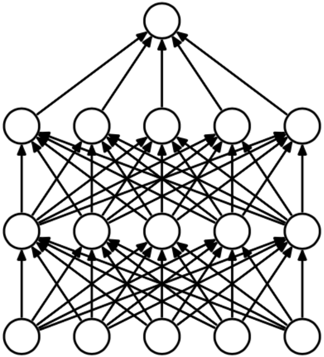
He initialization [He et al., 2015]

$$W \sim \mathcal{N}(0, \frac{2}{n_{in}})$$

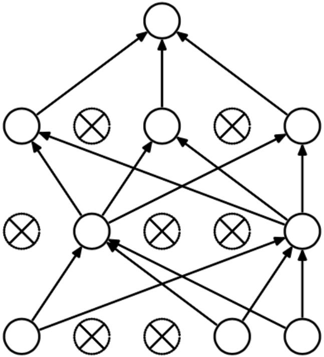
This is an actively research area and next great idea may come from you!

Dropout layer

"randomly set some neurons to zero in the forward pass" [Srivastava et al., 2014]

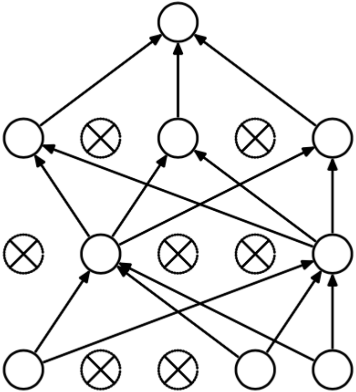


(a) Standard Neural Net



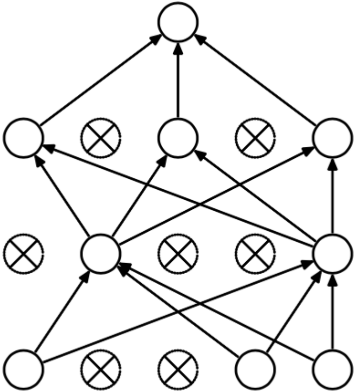
(b) After applying dropout.

Dropout layer



Forces the network to have a redundant representation.

Dropout layer



Another interpretation: Dropout is training a large ensemble of models.

Batch size

To recap, we have learned

1. gradient descent which uses all training examples to compute gradients.
2. stochastic gradient descent (SGD) which uses one training example to compute gradients.
3. mini-batch SGD which uses few training examples to compute gradients.

Batch size

To recap, we have learned

1. gradient descent which uses all training examples to compute gradients.
2. stochastic gradient descent (SGD) which uses one training example to compute gradients.
3. mini-batch SGD which uses few training examples to compute gradients.

In general, we can use a parameter batch size to compute the gradients from a few instances.

- N (the entire training data)
- 1 (a single instance)
- More common values: 16, 32, 64, 128

Upcoming Classes

- Friday: Neural network architectures other than feedforward neural networks
- Monday: In-class exercise of neural networks

References

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR. URL `http://proceedings.mlr.press/v9/glorot10a.html`.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014. URL `http://jmlr.org/papers/v15/srivastava14a.html`.