2022-10-24

Lecture 7. Convolutional Neural Networks, advanced technics

Lecture 7. Convolutional Neural Networks, advanced technics

Alex Avdyushenko

Kazakh-British Technical University

October 24, 2022

Hello, today we continue to talk about Convolutional Neural Networks, advanced optimization methods and architecture.

And again we will start with three questions for five minutes based on the materials of the previous lecture.

Five-minutes block

Please, write answers or send photos with them directly to me in private messages here in Teams, so that others cannot read your message. Last time, a lot of people did it, so I believe that this time you all will succeed.

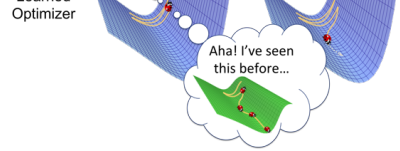
Lecture 7. Convolutional Neural Networks, advanced technics

Five-minutes block

Five-minutes block

- Define a convolution operation for neural networks
- Write down the main features of the AlexNet network
- Write the formulas for updating the weights in the Momentum method

Please, write answers or send photos with them directly to me in private messages here in Teams, so that others cannot read your message. Last time, a lot of people did it, so I believe that this time you all will succeed.



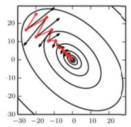
Let's look at this picture. In many ways, it reflects the entire content of today's lecture.

A very popular method for improving the convergence of SGD is the momentum accumulation method. In it, from a physical point of view, the derivative of the loss function becomes the acceleration of the change in model parameters, and not the speed as in classical SGD.

Momentum accumulation method [B.T.Polyak, 1964] — exponential moving average of the gradient over $\frac{1}{1-\gamma}$ last iterations:

$$\nu = \gamma\nu + (1-\gamma)\mathcal{L}_i'(w)$$

$$w = w - \eta\nu$$



A very popular method for improving the convergence of SGD is the momentum accumulation method. In it, from a physical point of view, the derivative of the loss function becomes the acceleration of the change in model parameters, and not the speed as in classical SGD.

Another interesting approach is the adaptive gradient method. You accumulate squares of gradient vector and then change the model parameters by the component-wise way.

AdaGrad

$$G = G + \mathcal{L}_i'(w) \odot \mathcal{L}_i'(w)$$
$$w = w - \eta_t \mathcal{L}_i'(w) \oslash \sqrt{G + \varepsilon}$$

η_t can be fixed, for example $\eta_t = 0.01$,
$\odot, \oslash$ — component-wise multiplication and division of vectors

Another interesting approach is the adaptive gradient method. You accumulate squares of gradient vector and then change the model parameters by the component-wise way.

Lecture 7. Convolutional Neural Networks, advanced technics

AdaGrad

AdaGrad

$$G = G + \mathcal{L}'_i(w) \odot \mathcal{L}'_i(w)$$
$$w = w - \eta_t \mathcal{L}'_i(w) \oslash \sqrt{G + \varepsilon}$$

η_t can be fixed, for example $\eta_t = 0.01$,
$\odot, \oslash$ — component-wise multiplication and division of vectors
Advantages:

- $\mathcal{L}'_i(w)$ — gradient on i-th object
- separate choice of adaptation for each direction
- is suitable for sparse data

Another interesting approach is the adaptive gradient method. You accumulate squares of gradient vector and then change the model parameters by the component-wise way.

Lecture 7. Convolutional Neural Networks, advanced technics

AdaGrad

AdaGrad

$$G = G + \mathcal{L}'_i(w) \odot \mathcal{L}'_i(w)$$
$$w = w - \eta_t \mathcal{L}'_i(w) \oslash \sqrt{G + \varepsilon}$$

η_t can be fixed, for example $\eta_t = 0.01$,
$\odot, \oslash$ — component-wise multiplication and division of vectors
Advantages:

- $\mathcal{L}'_i(w)$ — gradient on i-th object
- separate choice of adaptation for each direction
- is suitable for sparse data

Disadvantage:

- G is constantly increasing, which can cause learning to stop

Another interesting approach is the adaptive gradient method. You accumulate squares of gradient vector and then change the model parameters by the component-wise way.

The development of AdaGrad is RMSProp — optimization method, which you really have in modern libraries.

RMSProp (running mean square)

Equalization of rates of change of weights by moving average

$$G = \alpha G + (1 - \alpha)\mathcal{L}'_i(w) \odot \mathcal{L}'_i(w)$$
$$w = w - \eta_t \mathcal{L}'_i(w) \oslash \sqrt{G + \varepsilon}$$

η_t can be fixed, for example $\eta_t = 0.01$

The development of AdaGrad is RMSProp — optimization method, which you really have in modern libraries.

Lecture 7. Convolutional Neural Networks, advanced technics

RMSProp (running mean square)

RMSProp (running mean square)

Equalization of rates of change of weights by moving average

$$G = \alpha G + (1 - \alpha)\mathcal{L}'_i(w) \odot \mathcal{L}'_i(w)$$
$$w = w - \eta_t \mathcal{L}'_i(w) \oslash \sqrt{G + \varepsilon}$$

η_t can be fixed, for example $\eta_t = 0.01$
Advantages:

- same as AdaGrad
- and G does not grow uncontrollably

The development of AdaGrad is RMSProp — optimization method, which you really have in modern libraries.

Lecture 7. Convolutional Neural Networks, advanced technics

RMSProp (running mean square)

The development of AdaGrad is RMSProp — optimization method, which you really have in modern libraries.

RMSProp (running mean square)

Equalization of rates of change of weights by moving average

$$G = \alpha G + (1 - \alpha)\mathcal{L}'_i(w) \odot \mathcal{L}'_i(w)$$
$$w = w - \eta_t \mathcal{L}'_i(w) \oslash \sqrt{G + \varepsilon}$$

η_t can be fixed, for example $\eta_t = 0.01$
Advantages:

- same as AdaGrad
- and G does not grow uncontrollably

Disadvantages:

- at the very beginning of training G is averaged over zero previous values
- no moment count

Adam (adaptive momentum)

Adam is also one of modern optimization method, which you really may meet in modern applications.

Lecture 7. Convolutional Neural Networks, advanced technics

Adam (adaptive momentum)

Adam (adaptive momentum)

$$\nu = \gamma\nu + (1-\gamma)\mathcal{L}_i'(w) \qquad \hat{\nu} = \nu(1-\gamma^k)^{-1}$$
$$G = \alpha G + (1-\alpha)\mathcal{L}_i'(w) \odot \mathcal{L}_i'(w) \qquad \hat{G} = G(1-\alpha^k)^{-1}$$
$$w = w - \eta_t \hat{\nu} \oslash (\sqrt{\hat{G} + \varepsilon})$$

k — iteration number, $\gamma = 0.9, \alpha = 0.999, \varepsilon = 10^{-8}$

Adam is also one of modern optimization method, which you really may meet in modern applications.

Lecture 7. Convolutional Neural Networks, advanced technics

Adam (adaptive momentum)

Adam (adaptive momentum)

$$\nu = \gamma\nu + (1-\gamma)\mathcal{L}_i'(w) \qquad \hat{\nu} = \nu(1-\gamma^k)^{-1}$$
$$G = \alpha G + (1-\alpha)\mathcal{L}_i'(w) \odot \mathcal{L}_i'(w) \qquad \hat{G} = G(1-\alpha^k)^{-1}$$
$$w = w - \eta_t \hat{\nu} \oslash (\sqrt{\hat{G} + \varepsilon})$$

k — iteration number, $\gamma = 0.9, \alpha = 0.999, \varepsilon = 10^{-8}$
Advantages:

- are almost the same as RMSProp
- calibration $\hat{\nu}, \hat{G}$ increases ν, G on the first iterations
- is count of moment

Adam is also one of modern optimization method, which you really may meet in modern applications.

Lecture 7. Convolutional Neural Networks, advanced technics

Nadam (Nesterov-accelerated adaptive momentum)

Nadam (Nesterov-accelerated adaptive momentum)

Same formulas for $\nu, \hat{\nu}, G, \hat{G}$

$w = w - \eta_t \left(\gamma\hat{\nu} + \frac{1-\gamma}{1-\gamma^k}\mathcal{L}'_i(w)\right) \oslash (\sqrt{\hat{G} + \varepsilon})$

k — iteration number, $\gamma = 0.9, \alpha = 0.999, \varepsilon = 10^{-8}$

T. Dozat. Incorporating Nesterov Momentum into Adam. ICLR-2016.

And Nadam (Nesterov-accelerated adaptive momentum) — is modern modification of Adam method.

Source: http://www.denizyuret.com/2015/03/alec-radfords-animations-for.html

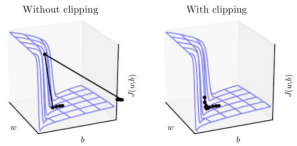
Let's look on example of convergence of considered optimization methods.

Lecture 7. Convolutional Neural Networks, advanced technics

Gradient exploding

There is typical problem in neural network optimization, it is gradient exploding.

Gradient exploding



You need to numpy.clip by value or by absolute value.

Source: https://neptune.ai/blog/understanding-gradient-clipping-and-how-it-can-fix-exploding-gradients-problem

Dropout is a method of random shutdowns of neurons. It has two stages: learning stage and application stage.

Dropout is a method of random shutdowns of neurons. It has two stages: learning stage and application stage.

Lecture 7. Convolutional Neural Networks, advanced technics

Regularization Methods

Regularization Methods

Dropout is a method of random shutdowns of neurons
Learning stage: make the gradient step $\mathcal{L}_i(w) \to \min_w$, disable the h-th neuron of the ℓ-th layer with probability p_ℓ:
$x_{ih}^{\ell+1} = \xi_h^\ell \sigma_h(\sum_j w_{jh} x_{ij}^\ell), \quad P(\xi_h^\ell = 0) = p_\ell$

Dropout is a method of random shutdowns of neurons. It has two stages: learning stage and application stage.

Lecture 7. Convolutional Neural Networks, advanced technics

Regularization Methods

Regularization Methods

Dropout is a method of random shutdowns of neurons
Learning stage: make the gradient step $\mathcal{L}_i(w) \to \min_w$, disable the h-th neuron of the ℓ-th layer with probability p_ℓ:
$x_{ih}^{\ell+1} = \xi_h^\ell \sigma_h(\sum_j w_{jh} x_{ij}^\ell), \quad P(\xi_h^\ell = 0) = p_\ell$
Application stage: turn on all neurons, but with a correction:
$x_{ih}^{\ell+1} = (1 - p_\ell)\sigma_h(\sum_j w_{jh} x_{ij}^\ell)$

Dropout is a method of random shutdowns of neurons. It has two stages: learning stage and application stage.

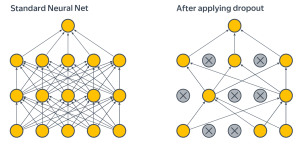
Lecture 7. Convolutional Neural Networks, advanced technics

Regularization Methods

Regularization Methods

Dropout is a method of random shutdowns of neurons
Learning stage: make the gradient step $\mathcal{L}_i(w) \to \min_w$, disable the h-th neuron of the ℓ-th layer with probability p_ℓ:
$x_{ih}^{\ell+1} = \xi_h^\ell \sigma_h(\sum_j w_{jh} x_{ij}^\ell), \quad P(\xi_h^\ell = 0) = p_\ell$

Application stage: turn on all neurons, but with a correction:
$x_{ih}^{\ell+1} = (1 - p_\ell)\sigma_h(\sum_j w_{jh} x_{ij}^\ell)$



Dropout is a method of random shutdowns of neurons. It has two stages: learning stage and application stage.

Lecture 7. Convolutional Neural Networks, advanced technics

Batch normalization

Batch normalization

$B = \{x_i\}$ — mini-batch of whole dataset
Averaging the $\mathcal{L}_i(w)$ gradients over the batch speeds up the convergence.
$B^\ell = \{u_i^\ell\}$ — vectors of x_i objects at the output of the ℓ-th layer

Let's go on to the next important regulzrization method. It is Batch normalization method.

Lecture 7. Convolutional Neural Networks, advanced technics

Batch normalization

Batch normalization

$B = \{x_i\}$ — mini-batch of whole dataset
Averaging the $\mathcal{L}_i(w)$ gradients over the batch speeds up the convergence.
$B^\ell = \{u_i^\ell\}$ — vectors of x_i objects at the output of the ℓ-th layer
1. Normalize each j-th component of the vector u_i^ℓ by the batch

$$\hat{u}_{ij}^\ell = \frac{u_{ij}^\ell - \mu_j}{\sqrt{\sigma_j^2 + \varepsilon}}; \quad \mu_j = \frac{1}{|B|}\sum_{x_i \in B} u_{ij}^\ell; \quad \sigma_j^2 = \frac{1}{|B|}\sum_{x_i \in B}(u_{ij}^\ell - \mu_j)^2$$

Let's go on to the next important regulzrization method. It is Batch normalization method.

Lecture 7. Convolutional Neural Networks, advanced technics

Batch normalization

Batch normalization

$B = \{x_i\}$ — mini-batch of whole dataset
Averaging the $\mathcal{L}_i(w)$ gradients over the batch speeds up the convergence.
$B^\ell = \{u_i^\ell\}$ — vectors of x_i objects at the output of the ℓ-th layer
1. Normalize each j-th component of the vector u_i^ℓ by the batch

$$\hat{u}_{ij}^\ell = \frac{u_{ij}^\ell - \mu_j}{\sqrt{\sigma_j^2 + \varepsilon}}; \quad \mu_j = \frac{1}{|B|}\sum_{x_i \in B} u_{ij}^\ell; \quad \sigma_j^2 = \frac{1}{|B|}\sum_{x_i \in B}(u_{ij}^\ell - \mu_j)^2$$

2. Add a linear layer with custom weights:

$$\tilde{u}_{ij}^\ell = \gamma_j^\ell \hat{u}_{ij}^\ell + \beta_j^\ell$$

Let's go on to the next important regulzrization method. It is Batch normalization method.

Lecture 7. Convolutional Neural Networks, advanced technics

Batch normalization

Let's go on to the next important regulzrization method. It is Batch normalization method.

Batch normalization

$B = \{x_i\}$ — mini-batch of whole dataset
Averaging the $\mathcal{L}_i(w)$ gradients over the batch speeds up the convergence.
$B^\ell = \{u_i^\ell\}$ — vectors of x_i objects at the output of the ℓ-th layer
1. Normalize each j-th component of the vector u_i^ℓ by the batch

$$\hat{u}_{ij}^\ell = \frac{u_{ij}^\ell - \mu_j}{\sqrt{\sigma_j^2 + \varepsilon}}; \quad \mu_j = \frac{1}{|B|}\sum_{x_i \in B} u_{ij}^\ell; \quad \sigma_j^2 = \frac{1}{|B|}\sum_{x_i \in B}(u_{ij}^\ell - \mu_j)^2$$

2. Add a linear layer with custom weights:

$$\tilde{u}_{ij}^\ell = \gamma_j^\ell \hat{u}_{ij}^\ell + \beta_j^\ell$$

3. Parameters $\gamma_j^\ell, \beta_j^\ell$ are configured by BackProp

Also of course very important to choose consistently the Initial Approximation of the Weights.

Lecture 7. Convolutional Neural Networks, advanced technics

Monitoring changes in variances

Equalization of variances of both neurons and gradients in different layers — Xavier initialization for tanh:

$$w_j \sim U\left[-\frac{6}{\sqrt{n_{in}+n_{out}}}, \frac{6}{\sqrt{n_{in}+n_{out}}}\right]$$

n_{in}, n_{out} — the number of neurons in the previous and current layers, respectively
The proof and details are in the SHAD textbook (in Russian yet).

In fact, the consistently rule is quite simple — you need to make sure that the variance of the layers is preserved both during the forward and backward pass through the neural network.

Equalization of variances of both neurons and gradients in different layers — Xavier initialization for tanh:

$$w_j \sim U\left[-\frac{6}{\sqrt{n_{in}+n_{out}}}, \frac{6}{\sqrt{n_{in}+n_{out}}}\right]$$

n_{in}, n_{out} — the number of neurons in the previous and current layers, respectively
The proof and details are in the SHAD textbook (in Russian yet).

Question
What to do in case of ReLU activation?

In fact, the consistently rule is quite simple — you need to make sure that the variance of the layers is preserved both during the forward and backward pass through the neural network.

Lecture 7. Convolutional Neural Networks, advanced technics

ResNet — Residual Net

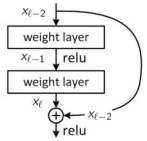
Great, now let's talk a bit about modern neural network architectures.

ResNet — Residual Net

Skip-connection of the ℓ layer with the previous $\ell - d$ layer:

$$x_\ell = \sigma(Wx_{\ell-1}) + x_{\ell-d}$$

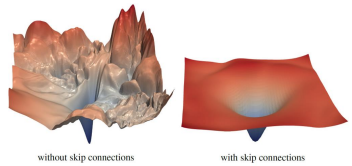
The ℓ layer learns not the new vector representation x_ℓ, but its increment $x_\ell - x_{\ell-d}$



Lecture 7. Convolutional Neural Networks, advanced technics

ResNet: visualization of loss function

ResNet: visualization of loss function



Hao Li et al. Visualizing the Loss Landscape of Neural Nets. 2018.

This is famous image from a paper that almost became a meme in deep learning :)

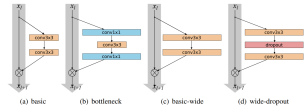
Lecture 7. Convolutional Neural Networks, advanced technics

└WideResNet — WRN

WideResNet — WRN

Disadvantages of ResNet

- is too deep — up to 1000 layers :)
- take a long time to train to SotA



- a simple WRN network of 16 layers defeated all ResNets
- WRN-40-4 (i.e. 40 layers and 4x width) trains 8 times faster than ResNet-1001

S. Zagoruyko, N. Komodakis Wide Residual Networks, 2017

And the final architecture for today's lecture WideResNet, which I recommend paying attention to when solving the competition from the second homework. It has already been posted on github and in teams.

Temporary page!

LaTeX was unable to guess the total number of pages correctly. A
was some unprocessed data that should have been added to the
this extra page has been added to receive it.
If you rerun the document (without altering it) this surplus page
away, because LaTeX now knows how many pages to expect for t
document.