4. Convolutional Neural networks

4.2. Operational elements of the CNN

Manel Martínez-Ramón
Meenu Ajith
Aswathy Rajendra Kurup

Introduction

- The basic CNN backpropagation is usually not sufficient to produce good results, due to the high complexity of most CNN structures. The techniques applied to the training of a CNN include:
  - The L_1 and L_2 regularization techniques explained in Module 1.
  - Other regularization techniques as dropout, early stopping, minibatch learning, data augmentation.
  - Data normalization techniques.
- The SDG algorithm is extended with the techniques so called momentum, Nesterov Accelerated Gradient, Adagrad, RMSProp, Adam and other similar optimizers.

L_1 and L_2 Regularization

- L_1 and L_2 regularization have been treated in Module 1. In the case of L_2, the regularization term is

$$J(\theta) = J_{ML}(\theta) + \frac{\lambda}{2}||\mathbf{W}^{(l)}||_F^2 \tag{1}$$

And its gradient is

$$\nabla_{\mathbf{W}^{(l)}} J(\theta) = \nabla_{\mathbf{W}^{(l)}} J_{ML}(\theta) + \lambda \mathbf{W}^{(l)} \tag{2}$$

- For the case of L_1, the regularization is

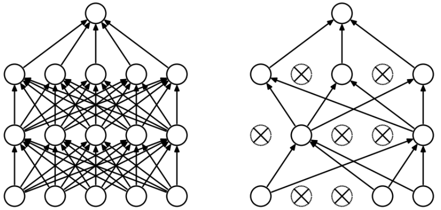
$$J(\theta) = J_{ML}(\theta) + \lambda||\mathbf{W}^{(l)}||_1 \tag{3}$$

with a gradient

$$\nabla_{\mathbf{W}^{(l)}} J(\theta) = \nabla_{\mathbf{W}^{(l)}} J_{ML}(\theta) + \lambda sign(\mathbf{W}^{(l)}) \tag{4}$$

Dropout

- Dropout is a regularization method consisting of disconnecting nodes at random during the training.
- This prevents nodes to co-adapt.


(a) Standard Neural Net (b) After applying dropout.

- During training, keep nodes randomly with probability p (usually $p > 0.5$) at each minibatch training.
- During test, use all weights multiplied by p.

- Dropout is usually combined with a max-norm regularization that forces the norm of the weights to be under a given maximum value.

Data augmentation

- Generating new training samples to increase its diversity.
- For images, data augmentation is done by geometric transformations on data: Cropping, flipping, zooming, translations, and blurring specific pixels.
- Lighting biases are fixed by color space transformations.
- Deep learning-based data augmentation uses Generative Adversarial Networks (GAN) and feature space augmentations.

Data augmentation from the CIFAR10 dataset.

Optimizers
SDG with momentum

The momentum in an optimizer consists of a fraction of the previous gradient descent.

$$\mathbf{V}_k = \gamma \mathbf{V}_{k-1} - \mu \nabla_{\mathbf{W}} J(\mathbf{W}_k) \tag{5}$$

Parameter γ controls the quantity of momentum in the descent.

$$\mathbf{W}_{k+1} = \mathbf{W}_0 + \sum_{k'=0}^{k} \mathbf{V}_{k'} = \mathbf{W}_k + \mathbf{V}_k \tag{6}$$

$\mathbf{V}$ plays the role of a velocity vector.

Optimizers

Nesterov Accelerated Gradient

- With momentum optimization, if the particle arrives at the minimum, it will keep moving with a vanishing oscillatory movement.
- The Nesterov accelerated gradient descent looks at the gradient one step ahead. The update of the weight vector would be

$$\tilde{\mathbf{W}}_{k+1} = \mathbf{W}_k + \mathbf{V}_k \tag{7}$$

- The update equations are

$$\begin{aligned} \mathbf{V}_k &= \gamma \mathbf{V}_{k-1} - \mu \nabla_{\mathbf{W}} J(\tilde{\mathbf{W}}_k) \\ \mathbf{W}_{k+1} &= \mathbf{W}_k + \mathbf{V}_k \end{aligned} \tag{8}$$

Optimizers
Adagrad

- Adapts the learning rate to each one of the parameters as a function of the norm of the gradient at each one of these parameters.
- It first computes the accumulated square value of each component of the gradient

$$\mathbf{G}_k = \mathbf{G}_{k-1} + \nabla_{\mathbf{W}} J(\mathbf{W}_k) \odot \nabla_{\mathbf{W}} J(\mathbf{W}_k) \tag{9}$$

- Then, it updates each one of the parameters $w_{i,k}$ in $\mathbf{w}_k$ as

$$w_{i,k+1} = w_{i,k} - \frac{\mu}{\sqrt{g_{i,k}} + \varepsilon} \left[\nabla_{\mathbf{W}} J(\mathbf{W}_k)\right]_i \tag{10}$$

where $[\nabla_{\mathbf{W}} J(\mathbf{W}_k)]_i = \frac{d}{dw_i} J(\mathbf{W}_k)$ is the i^{th} element of the gradient vector and ε is a small number that is used for numerical stability.

Optimizers
RMSProp

- The AdaGrad algorithm has a learning rate that decreases monotonically with time, which will end up in a learning rate that tends to zero.
- The root mean square propagation (RMSProp) algorithm forgets about the squared gradients of remote time instants through an exponential decay

$$\mathbf{G}_k = \gamma \mathbf{G}_{k-1} + (1-\gamma)\, \nabla_{\mathbf{W}} J(\mathbf{W}_k) \odot \nabla_{\mathbf{W}} J(\mathbf{W}_k) \qquad (11)$$

- If $\gamma = 1$ the algorithm forgets everything about the past gradients. If $\gamma = 0$, the algorithm is identic to the AdaGrad one.

Optimizers

Adam

The Adam (Adaptive Momentum Estimation) algorithm is a combination of the previous two ones.

$$\mathbf{V}_k = \beta_1 \mathbf{V}_{k-1} - (1-\beta_1) \nabla_{\mathbf{W}} J(\mathbf{W}_k) \tag{12}$$

$$\mathbf{G}_k = \beta_2 \mathbf{G}_{k-1} + (1-\beta_2)\, \nabla_{\mathbf{W}} J(\mathbf{W}_k) \odot \nabla_{\mathbf{W}} J(\mathbf{W}_k) \tag{13}$$

$$\begin{aligned} \tilde{\mathbf{V}}_k &= \frac{\mathbf{V}_k}{1-\beta_1^k} \\ \tilde{\mathbf{G}}_k &= \frac{\mathbf{G}_k}{1-\beta_2^k} \end{aligned} \tag{14}$$

Each element of the weight vector $\mathbf{w}_k$ is updated as

$$w_{i,k+1} = w_{i,k} - \mu \frac{v_{i,k}}{\sqrt{g_{i,k}} + \varepsilon} \tag{15}$$