

backpropagation

Author and editors

Alexander Jung*, Department of Computer Science, Aalto University, Espoo, Finland

Editors: Konstantina Olioumtsevs, Ekkehard Schnoor, Aalto University, Espoo, Finland

*Corresponding author: alex.jung@aalto.fi

Abstract

Backpropagation is an algorithm that computes the gradient of an objective function with respect to the model parameters of an artificial neural network (ANN). It runs in two passes. The forward pass sends a batch of data points through the ANN and evaluates the loss of the resulting predictions. The backward pass then applies the chain rule from the output back to the input, computing on the way the partial derivative with respect to every weight. Obtaining all of them this way costs a small multiple of one forward pass, where perturbing each weight separately would cost one pass per weight. Backpropagation is not itself an iteration: it yields one gradient, which a gradient step then consumes. Every training step of a deep net uses it.

Definition

An image classifier reads the features of a photograph, passes them through layers of neurons, and delivers at its output a prediction of the label. Training it requires the partial derivative of the loss with respect to each of its millions of weights. Obtaining those one at a time, by perturbing a single weight and evaluating the loss again, would cost one pass through the network per weight. Backpropagation delivers all of them in two passes (Rumelhart et al., 1986; Goodfellow et al., 2016). It is an algorithm that computes the gradient $\nabla_{\mathbf{w}} f(\mathbf{w})$ of an objective function $f(\mathbf{w})$ with respect to the model parameters $\mathbf{w}$ of an artificial neural network (ANN), which are the weights of its layers collected into one vector. It applies the chain rule of calculus to evaluate the partial derivatives of f one layer at a time (see Fig. 1).

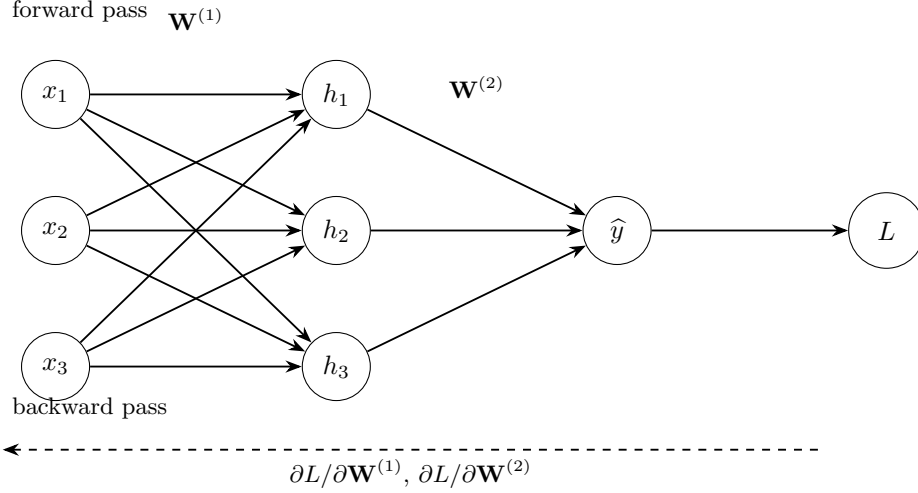


Figure 1: Backpropagation through an ANN with one hidden layer. The solid arrows are the forward pass: the features x_1, x_2, x_3 reach the neurons h_1, h_2, h_3 through the weights $\mathbf{W}^{(1)}$, then the output $\hat{y}$ through $\mathbf{W}^{(2)}$, and the loss L compares that output to the label. The dashed arrow is the backward pass, which runs the other way and yields the partial derivatives of L with respect to the weights of both layers

In the forward pass, a batch of data points travels through the ANN: each layer applies its weights and its activation function, producing predictions at the output, and the loss function compares those predictions to the true labels. In the backward pass, the partial derivatives of the loss with respect to the weights of each layer are computed recursively from the output back to the input, which yields the whole gradient $\nabla_{\mathbf{w}} f(\mathbf{w})$. Obtaining every partial derivative this way costs a small multiple of one forward pass, instead of one pass per weight.

Backpropagation is not itself an iteration: one forward and one backward pass yield one gradient. The iteration is gradient descent (GD), which consumes that gradient in a gradient step

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \nabla f(\mathbf{w}^{(t)}),$$

with step size η , and calls backpropagation again at the new model parameters. The objective function $f(\mathbf{w})$ is the average loss on a training set, while the goal of training is a small loss on data points outside that training set. Backpropagation makes the search over $\mathbf{w}$ affordable; it does not close that gap.

Every training step of a deep net uses backpropagation, once per batch. In image classification, for example, it computes the partial derivatives needed to update millions of weights spread over dozens of layers.

Cross References

ANN, deep net, gradient, GD, gradient step, layer, weight, model parameter, activation function, loss function

References

1. Goodfellow et al. (2016). Deep Learning. MIT Press, Cambridge, MA, USA.
2. Rumelhart et al. (1986). Learning Representations by Back-Propagating Errors. Nature, 323, pp. 533–536.