Backpropagation

Yao Zhang

The guy is a populace

Mostly based on Thomas Hofmann's lecture in ETH

https://zhims.github.io/

Dec 3, 2019

Gradient Descent

Learning in neural networks = gradient-based optimization (with very few exceptions).

Definition 1 (Gradient)

Gradient of objective with regard to parameters θ

$$\nabla_\theta = \left(\frac{\partial \mathcal{R}}{\partial \theta_1}, ..., \frac{\partial \mathcal{R}}{\partial \theta_d} \right)^T \tag{1}$$

Definition 2 (Steepest descent and stochastic gradient decent)

Steepest descent and stochastic gradient decent

$$\theta(t+1) \leftarrow \theta(t) - \eta \nabla_\theta \mathcal{R}(\mathcal{S}) \tag{2}$$

1. here $t = 0, 1, 2, ...$ is an iteration index
2. $\mathcal{S}$ =all training data $\Rightarrow$ steepest descent
3. $\mathcal{S}$ = mini batch of data $\Rightarrow$ SGD

Gradient Computation via Backpropagation

Computational challenge: how to compute $\nabla_\theta \mathcal{R}$?
Exploit compositional structure of network = backpropagation
Basic steps:

1. perform a forward pass (for given training input x) to compute activations for all units
2. compute gradient of $\mathcal{R}$ w.r.t. output layer activations (for given target y)
3. iteratively propagate activation gradient information from outputs to inputs
4. compute local gradients of activations w.r.t weights

Backpropagation in Plain English

1. How do changes in the output layer activities change the objective?
    - depends on choice of objective
2. How does the activity of a parent unit influence the activity of each of its child units (in DAG)?
    - layer structure $\Rightarrow$ concurrently between subsequent layers
3. Propagate influence information through reverse DAG
    - details are implied by chain rule of differentiation
4. What is the effect of a change of an incoming weight on the activity of a unit?
    - can only change activities (given x) by modifying weights

Chain Rule

Compositional of functions $\Rightarrow$ use of chain rule

Proposition 1 (Chain Rule)

$$(f \circ g)' = (f' \circ g) \cdot g' \tag{3}$$

or equivalently with formal variables

$$\frac{d\,(f \circ g)}{dx}\Big|_{x=x_0} = \frac{df}{dz}\Big|_{z=g(x_0)} \cdot \frac{dg}{dx}\Big|_{x=x_0} \tag{4}$$

Jacobi Matrix

Vector-valued function (map) $F : \mathcal{R}^n \to \mathcal{R}^m$: each component function has gradient $\nabla F_i \in \mathcal{R}^n$, $i \in [1 : m]$

Definition 3 (Jacobi matrix)

$$J_F \triangleq \begin{pmatrix} \nabla^T F_1 \\ \nabla^T F_2 \\ \vdots \\ \nabla^T F_m \end{pmatrix} = \begin{pmatrix} \frac{\partial F_1}{\partial x_1} & \frac{\partial F_1}{\partial x_2} & \cdots & \frac{\partial F_1}{\partial x_n} \\ \frac{\partial F_2}{\partial x_1} & \frac{\partial F_2}{\partial x_2} & \cdots & \frac{\partial F_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial F_m}{\partial x_1} & \frac{\partial F_m}{\partial x_2} & \cdots & \frac{\partial F_m}{\partial x_n} \end{pmatrix} \in \mathbb{R}^{m \times n} \tag{5}$$

derivative of outputs with regard to inputs, i.e. $(J_F)_{ij} = \frac{\partial F_i}{\partial x_j}$.

Jacobin Matrix Chain Rule

Vector-valued functions $G : \mathbb{R}^n \to \mathbb{R}^q,\ H : \mathbb{R}^q \to \mathbb{R}^m, F \triangleq H \circ G$.
Componentwise rule

$$\frac{\partial F_i}{\partial x_j}\Big|_{x=x_0} = \frac{\partial (H \circ G)_i}{\partial x_j}\Big|_{x=x_0} = \sum_{k=1}^{q} \frac{\partial H_i}{\partial z_k}\Big|_{z=G(x_0)} \cdot \frac{\partial G_k}{\partial x_j}\Big|_{x=x_0} \tag{6}$$

Lemma 1 (Jacobi matrix chain rule)

$$J_{H \circ G}\big|_{x=x_0} = J_H\big|_{z=G(x_0)} \cdot J_G\big|_{x=x_0} \tag{7}$$

Function Composition

Special case: composition of a map with a function

$$G : \mathbb{R}^n \to \mathbb{R}^m, \quad h : \mathbb{R}^m \to \mathbb{R}, \quad h \circ G : \mathbb{R}^n \to \mathbb{R} \tag{8}$$

$x \in \mathbb{R}^n$, use more intuitive variable notation

$$x \overset{G}{\mapsto} y \overset{h}{\mapsto} z \in \mathbb{R} \tag{9}$$

Then

$$\nabla_x^T z = \nabla_y^T z \cdot J_G, \quad \frac{\partial z}{\partial x_i} = \sum_j \frac{\partial y_j}{\partial x_i} \frac{\partial z}{\partial y_j} \tag{10}$$

Warning: Notation!

We have a lot of indices!

- index of a layer: put as a superscript
- index of a dimension of a vector: put as a subscript
- shorthand for layer activations

$$\begin{aligned} x^l &\triangleq \left(F^l \circ \cdots \circ F^1\right)(x) \in \mathbb{R}^{m_l} \\ x_i^l &\in \mathbb{R} : \ activation \ \ of \ \ i - th \ \ unit \ \ in \ \ layer \ \ l \end{aligned} \tag{11}$$

- index of a data point, omitted where possible, rectangular brackets $(x[i], y[i])$

Composition of multiple maps with a final cost function

$$
\begin{aligned}
F = F^L \circ \cdots \circ F^1 :\ \ \mathbb{R}^n \to \mathbb{R}^m \\
x = x^0 \overset{F^1}{\mapsto} x^1 \overset{F^2}{\mapsto} x^2 \mapsto \cdots \overset{F^L}{\mapsto} x^L = \nu \mapsto \ell(y, \nu)
\end{aligned}
\tag{12}
$$

Proposition 2 (Activity Backpropagation)

$$
e^L \triangleq \nabla_\nu^T \mathcal{R}, \ \ e^l \triangleq \nabla_{x^l}^T \mathcal{R} = e^L \cdot J_{F^L} \cdots \ J_{F^{l+1}} = e^{l+1} \cdot J_{F^{l+1}} \tag{13}
$$

Compute activity gradients is backward order via successive multiplication with Jacobians. Backpropagation of error terms e^l.
Linear nrtwork in reversed direction with "activities" e^l.

Jacobian Matrix: Ridge Functions

How does a Jacobian matrix for a ridge function look like?

$$x^l = F^l\left(x^{l-1}\right) = \sigma\left(W^l x^{l-1} + b^l\right) \tag{14}$$

Hence (assuming differentiability of σ):

$$\frac{\partial x_i^l}{\partial x_j^{l-1}} = \sigma'\left(\left\langle w_i^l, x^{l-1}\right\rangle + b_i^l\right) w_{ij}^l \triangleq \widetilde{w}_{ij}^l \tag{15}$$

and thus simply

$$J_{F^l} = \widetilde{W}^l \tag{16}$$

1. for ReLU $\widetilde{w}_{ij}^l \in \left\{0, w_{ij}^l\right\} \Rightarrow \widetilde{W}^l =$ sparsified matrix

Loss Function (Negative) Gradients

Quadratic loss

$$-\nabla_\nu \ell(y,\nu) = -\nabla_\nu \frac{1}{2}\|y-\nu\|^2 = y-\nu \tag{17}$$

Multivariate logistic loss

$$\begin{aligned}
-\frac{\partial \ell(y,\nu)}{\partial z_y} &= \frac{\partial}{\partial z_y}\left[z_{y^*} - \log\sum_i e^{z_i}\right] \\
&= \delta_{yy^*} - \frac{e^{z_y}}{\sum_i e^{z_i}} \\
&= \delta_{yy^*} - p(y|x)
\end{aligned} \tag{18}$$

From Activations to Weights

How can we get from gradients w.r.t. activations to gradients w.r.t. weights? Easily!
Need to apply chain rule one more time- locally:

$$\frac{\partial \ell}{\partial w_{ij}^l} = \frac{\partial \ell}{\partial x_i^l} \cdot \frac{\partial x_i^l}{\partial w_{ij}^l} = \underbrace{\frac{\partial \ell}{\partial x_i^l}}_{backprop} \cdot \underbrace{\sigma' \left(\left\langle w_i^l, x^{l-1} \right\rangle + b_i^l \right)}_{sensitivity \ \ of \ i-th \ unit} \cdot \underbrace{x_j^{l-1}}_{j-th \ unit \ activity} \tag{19}$$

$$\frac{\partial \ell}{\partial b_i^l} = \frac{\partial \ell}{\partial x_i^l} \cdot \frac{\partial x_i^l}{\partial b_i^l} = \frac{\partial \ell}{\partial x_i^l} \cdot \sigma' \left(\left\langle w_i^l, x^{l-1} \right\rangle + b_i^l \right) \cdot 1$$

- each weight/bias influences exactly one unit
- can "reshape" gradient into matrix/tensor form

Specialized Programming Languages: Theano

Symbolic representation of mathematical expressions.

Access to full computation graph (stability, optimization).

Symbolic differentiation.

[Bergstra 2015]

reference

J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley and Y. Bengio (2010)

Theano: A CPU and GPU Math Compiler in Python

9th Annual Python In Science Conference (SciPy 2010)

Last But Not Least

Thank you all of you! –Yao