TOPIC 3

Optimality, convexity, and gradient descent

5 Automatic differentiation: theory

Course: Math 535 (http://www.math.wisc.edu/~roch/mmids/) - Mathematical Methods in Data Science (MMiDS)
Author: Sebastien Roch (http://www.math.wisc.edu/~roch/), Department of Mathematics, University of Wisconsin-Madison
Updated: Nov 12, 2020

We develop the basic mathematical foundations of automatic differentiation. We restrict ourselves to a special setting: multi-layer progressive functions. Many important classifiers take the form of a sequence of compositions where parameters are specific to each layer of composition. We show how to systematically apply the *Chain Rule* to such functions. We give examples in the next section.

Before delving into the details, we make two important remarks:

(1) A classifier h takes an input in $\mathbb{R}^d$ and predicts one of K possible labels. It will be convenient for reasons that will become clear below to use one-hot encoding (https://en.wikipedia.org/wiki/One-hot) of the labels. That is, we encode label i as the K-dimensional vector $\mathbf{e}_i$. Here, as usual, $\mathbf{e}_i$ the canonical basis of $\mathbb{R}^K$, i.e., the vector with a 1 in entry i and a 0 elsewhere. Furthermore, we allow the output of the classifier to be a probability distribution over the labels $\{1, \ldots, K\}$, that is, a vector in

$$\Delta_K = \left\{ (p_1, \ldots, p_K) \in [0,1]^K \,:\, \sum_{k=1}^K p_k = 1 \right\}.$$

(2) One source of confusion here is that, while it is natural to define a classifier h as a function of the input data $\mathbf{x} \in \mathbb{R}^d$, when fitting the data we are ultimately interested in thinking of h as a function of the parameters $\mathbf{w} \in \mathbb{R}^r$ that need to be adjusted -- over a fixed dataset. Hence in this section, we emphasize thoughout in the notation that $\mathbf{x}$ is fixed while $\mathbf{w}$ is variable.

5.1 Progressive functions

The general model is defined as follows. Throughout this section, we use the following notation: for two vectors $\mathbf{v}^1 \in \mathbb{R}^{a_1}$ and $\mathbf{v}^2 \in \mathbb{R}^{a_2}$, the concatenation of $\mathbf{v}^1$ and $\mathbf{v}^2$ as a vector in $\mathbb{R}^{a_1+a_2}$ is denoted by $(\mathbf{v}^1; \mathbf{v}^2)$. For example, if $\mathbf{v}^1 = (1, 2)^T$ and $\mathbf{v}^2 = (-1, -3, -5)^T$, then $(\mathbf{v}^1; \mathbf{v}^2) = (1, 2, -1, -3, -5)^T$. More generally, $(\mathbf{v}^1; \dots; \mathbf{v}^i)$ is the concatenation of vectors $\mathbf{v}^1, \dots, \mathbf{v}^i$.

We have L layers. Layer i is defined by a continuously differentiable function g_i which takes two vector-valued inputs: a vector of parameters $\mathbf{w}^i \in \mathbb{R}^{r_i}$ specific to the i-th layer and a vector of inputs $\mathbf{z}^i \in \mathbb{R}^{n_i}$ which is fed from the $i-1$-th layer

$$g_i : \mathbb{R}^{r_i+n_i} \to \mathbb{R}^{n_{i+1}}.$$

The output of g_i is a vector in $\mathbb{R}^{n_{i+1}}$ which is passed to the $i+1$-th layer as input.

We denote by $\mathbf{z}^1 = \mathbf{x} \in \mathbb{R}^d$ the input to the first layer. For $i = 1, \dots, L$, let

$$\mathbf{w}^{\leq i} = (\mathbf{w}^i; \dots; \mathbf{w}^1) \in \mathbb{R}^{r_i+\dots+r_1}$$

be the concatenation of the parameters from the first i layers as a vector in $\mathbb{R}^{r_i+\dots+r_1}$. Note that the layer-specific parameters are ordered backwards in $\mathbf{w}^{\leq i}$. Then the output of layer i, as a function of the parameters, is the composition

$$\mathcal{O}_{i,\mathbf{x}}(\mathbf{w}^{\leq i}) = g_i(\mathcal{I}_{i,\mathbf{x}}(\mathbf{w}^{\leq i})) \in \mathbb{R}^{n_{i+1}},$$

where the input to layer i (both layer-specific parameters and output of previous layer), again as a function of the parameters, is

$$\mathcal{I}_{i,\mathbf{x}}(\mathbf{w}^{\leq i}) = \left(\mathbf{w}^i; \mathcal{O}_{i-1,\mathbf{x}}(\mathbf{w}^{\leq i-1})\right) \in \mathbb{R}^{r_i+n_i}.$$

Letting $\mathbf{w} = \mathbf{w}^{\leq L}$, the final output is

$$h_{\mathbf{x}}(\mathbf{w}) = \mathcal{O}_{L,\mathbf{x}}(\mathbf{w}^{\leq L}).$$

Expanding out the composition, this can be written alternatively as

$$h_{\mathbf{x}}(\mathbf{w}) = g_L(\mathbf{w}^L; g_{L-1}(\mathbf{w}^{L-1}; \cdots g_2(\mathbf{w}^2; g_1(\mathbf{w}^1; \mathbf{x})) \cdots)).$$

As a final step, we have a loss function $\ell_{\mathbf{y}} : \mathbb{R}^{n_{L+1}} \to \mathbb{R}$ which takes as input the output of the last layer and quantifies the fit to the given label $\mathbf{y} \in \Delta_K$. We will see some example below. The final function is then

$$f_{\mathbf{x},\mathbf{y}}(\mathbf{w}) = \ell_{\mathbf{y}}(h_{\mathbf{x}}(\mathbf{w})) \in \mathbb{R}.$$

We seek to compute the gradient of $f_{\mathbf{x},\mathbf{y}}(\mathbf{w})$ with respect to the parameters $\mathbf{w}$ in order to apply a gradient descent method.

5.2 Jacobians

Recall that the key insight from the *Chain Rule* is that to compute the gradient of a composition such as $h_{\mathbf{x}}(\mathbf{w})$ -- no matter how complex -- it suffices to *separately* compute the Jacobians of the intervening functions and then to take *matrix products*. In this section, we compute the necessary Jacobians in the progressive case.

The basic composition step is

$$\mathcal{O}_{i,\mathbf{x}}(\mathbf{w}^{\leq i}) = g_i(\mathcal{I}_{i,\mathbf{x}}(\mathbf{w}^{\leq i})).$$

Applying the *Chain Rule* we get

$$J_{\mathcal{O}_{i,\mathbf{x}}}(\mathbf{w}^{\leq i}) = J_{g_i}(\mathcal{I}_{i,\mathbf{x}}(\mathbf{w}^{\leq i}))\, J_{\mathcal{I}_{i,\mathbf{x}}}(\mathbf{w}^{\leq i}).$$

The Jacobian of

$$\mathcal{I}_{i,\mathbf{x}}(\mathbf{w}^{\leq i}) = \left(\mathbf{w}^i; \mathcal{O}_{i-1,\mathbf{x}}(\mathbf{w}^{\leq i-1})\right)$$

has a block structure

$$J_{\mathcal{I}_{i,\mathbf{x}}}(\mathbf{w}^{\leq i}) = \begin{pmatrix} I_{r_i \times r_i} & 0 \\ 0 & J_{\mathcal{O}_{i-1,\mathbf{x}}}(\mathbf{w}^{\leq i-1}) \end{pmatrix}$$

since its second component, $\mathcal{O}_{i-1,\mathbf{x}}(\mathbf{w}^{\leq i-1})$, does not depend on $\mathbf{w}^i$. Recall that the layer parameters are ordered backwards in $\mathbf{w}^{\leq i} = (\mathbf{w}^i; \ldots; \mathbf{w}^1)$.

We partition the Jacobian of g_i likewise, that is, we divide it into those columns corresponding to partial derivatives with respect to $\mathbf{w}^i$ (denoted A_i) and with respect to $\mathbf{z}^i$ (denoted B_i)

$$J_{g_i}(\mathcal{I}_{i,\mathbf{x}}(\mathbf{w}^{\leq i})) = \begin{pmatrix} A_i & B_i \end{pmatrix}.$$

Plugging back above we get

$$J_{\mathcal{O}_{i,\mathbf{x}}}(\mathbf{w}^{\leq i}) = \begin{pmatrix} A_i & B_i \end{pmatrix} \begin{pmatrix} I_{r_i \times r_i} & 0 \\ 0 & J_{\mathcal{O}_{i-1,\mathbf{x}}}(\mathbf{w}^{\leq i-1}) \end{pmatrix}.$$

This leads to the fundamental recursion

$$J_{\mathcal{O}_{i,\mathbf{x}}}(\mathbf{w}^{\leq i}) = \begin{pmatrix} A_i & B_i\, J_{\mathcal{O}_{i-1,\mathbf{x}}}(\mathbf{w}^{\leq i-1}) \end{pmatrix}$$

from which the Jacobian of $h_{\mathbf{x}}(\mathbf{w})$ can be computed.

Finally, using the *Chain Rule* again

$$\nabla f_{\mathbf{x},\mathbf{y}}(\mathbf{w})^T = J_{f_{\mathbf{x},\mathbf{y}}}(\mathbf{w}) = J_{\ell_{\mathbf{y}}}(h_{\mathbf{x}}(\mathbf{w}))\, J_{h_{\mathbf{x}}}(\mathbf{w}) = \nabla \ell_{\mathbf{y}}(h_{\mathbf{x}}(\mathbf{w}))^T\, J_{h_{\mathbf{x}}}(\mathbf{w}).$$

5.3 Forward and backward propagation

We take advantage of the recursion above to compute the gradient of $h_{\mathbf{x}}$. There are two ways a doing this. Applying the recursion directly is one of them, but it requires many matrix-matrix products. Instead, one can also run the recursion backwards. The latter approach can be much faster because, as we detail next, it involves instead matrix-vector products.

Start with the equation

$$\nabla f_{\mathbf{x},\mathbf{y}}(\mathbf{w}) = \boldsymbol{J}_{h_{\mathbf{x}}}(\mathbf{w})^T \, \nabla \ell_{\mathbf{y}}(h_{\mathbf{x}}(\mathbf{w}))$$

which was obtained by taking the transpose of the equation we derived for the gradient of f. Note that $\nabla \ell_{\mathbf{y}}(h_{\mathbf{x}}(\mathbf{w}))$ is a vector -- not a matrix. Then expand the matrix $\boldsymbol{J}_{h_{\mathbf{x}}}(\mathbf{w})$ using the recursion above

$$\begin{aligned}
\nabla f_{\mathbf{x},\mathbf{y}}(\mathbf{w}) &= \boldsymbol{J}_{h_{\mathbf{x}}}(\mathbf{w})^T \, \nabla \ell_{\mathbf{y}}(h_{\mathbf{x}}(\mathbf{w})) \\
&= \boldsymbol{J}_{\mathcal{O}_{L,\mathbf{x}}}(\mathbf{w}^{\leq L})^T \, \nabla \ell_{\mathbf{y}}(h_{\mathbf{x}}(\mathbf{w})) \\
&= \begin{pmatrix} A_L & B_L \, \boldsymbol{J}_{\mathcal{O}_{L-1,\mathbf{x}}}(\mathbf{w}^{\leq L-1}) \end{pmatrix}^T \nabla \ell_{\mathbf{y}}(h_{\mathbf{x}}(\mathbf{w})) \\
&= \begin{pmatrix} \boldsymbol{J}_{\mathcal{O}_{L-1,\mathbf{x}}}(\mathbf{w}^{\leq L-1})^T B_L^T & A_L^T \end{pmatrix} \nabla \ell_{\mathbf{y}}(h_{\mathbf{x}}(\mathbf{w})) \\
&= \begin{pmatrix} \boldsymbol{J}_{\mathcal{O}_{L-1,\mathbf{x}}}(\mathbf{w}^{\leq L-1})^T \, [B_L^T \, \nabla \ell_{\mathbf{y}}(h_{\mathbf{x}}(\mathbf{w}))] & [A_L^T \, \nabla \ell_{\mathbf{y}}(h_{\mathbf{x}}(\mathbf{w}))] \end{pmatrix}.
\end{aligned}$$

Continuing by induction gives an alternative formula for the gradient of $f_{\mathbf{x},\mathbf{y}}$. The key is that both $B_L^T \, \nabla \ell_{\mathbf{y}}(h_{\mathbf{x}}(\mathbf{w}))$ and $A_L^T \, \nabla \ell_{\mathbf{y}}(h_{\mathbf{x}}(\mathbf{w}))$ are matrix-vector products, and that pattern persists at next levels of recursion.

We give the full algorithm, in two passes, next.

In the forward pass, or forward propagation step, we compute the following:

Initialization:

$$\mathbf{z}^1 := \mathbf{x}$$

Forward layer loop: For $i = 1, \ldots, L$:

$$\begin{aligned}
\mathbf{z}^{i+1} &:= g_i(\mathbf{w}^i; \mathbf{z}^i) \\
\begin{pmatrix} A_i & B_i \end{pmatrix} &:= \boldsymbol{J}_{g_i}(\mathbf{w}^i; \mathbf{z}^i)
\end{aligned}$$

Loss:

$$\begin{aligned}
\mathbf{z}^{L+2} &:= \ell_{\mathbf{y}}(\mathbf{z}^{L+1}) \\
\mathbf{q}^{L+1} &:= \nabla \ell_{\mathbf{y}}(\mathbf{z}^{L+1}).
\end{aligned}$$

In the backward pass, or backpropagation step, we compute the following:

Backward layer loop: For $i = L, \ldots, 1$:

$$\begin{aligned} \mathbf{p}^i &:= A_i^T \mathbf{q}^{i+1} \\ \mathbf{q}^i &:= B_i^T \mathbf{q}^{i+1} \end{aligned}$$

Output:

$$\nabla f_{\mathbf{x},\mathbf{y}}(\mathbf{w}) = (\mathbf{p}^L; \ldots; \mathbf{p}^1).$$

Note that we do not in fact need to compute B_1 and $\mathbf{q}^1$.