Topics:

- ...

Derivative properties

- $\left(k \cdot f(x)\right)' = \frac{k \cdot f(x_2) - k \cdot f(x_1)}{x_2 - x_1} = k \frac{f(x_2) - f(x_1)}{x_2 - x_1} = kf'(x)$
- $\left(g(x) + f(x)\right)' = \frac{g(x_2) + f(x_2) - (g(x_1) + f(x_1))}{x_2 - x_1} = \frac{g(x_2) - g(x_1) + f(x_2) - f(x_1)}{x_2 - x_1} = g' + f'$
- $(\sum \ k_i f_i(x))' = \sum \ k_i f_i{}'(x))$

Gradient Properties

- Gradient is a vector which consist of partial derivatives

- Similar to $\left(\sum k_i f_i(x)\right)' = \sum k_i f_i'(x)$)

- Similar to $\nabla \sum k_i f_i(x) = \sum k_i \cdot \nabla f_i(x)$

How numerially evaluate derivative?

- Approach (1) for first derivative

$$f'(x) = \frac{f(x+h) - f(x)}{h} + O(h)$$
$$O(h) = -\frac{h}{2} f''(x) + \ldots$$

- Approach (2) for first derivative

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} + O(h^2)$$
$$O(h^2) = -\frac{h^2}{6} f'''(x) + \ldots$$

How numerially second derivative?

Numerically we can evaluate second derivative in the following way:

$$f''(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} + O(h^2)$$

$$O(h^2) = -\frac{h^2}{12} f''''(x) + \ldots$$

Derivative to compute partial derivatives

For single layer network and for single observation we have

$$\mathrm{r}_i = y_i - \overset{\frown}{F}(x_i)$$

$$L = \frac{1}{2}\left(y_i - \overset{\frown}{F}(x_i)\right)^2 = \frac{1}{2}r^2$$

$$\frac{dL}{dr} = \frac{2}{2}r = r = (y_i - \overset{\frown}{F}(x_i))$$

$$\overset{\frown}{F}(x; \color{red}{a, b}) = \sum_{m=0}^{M} b_m \cdot S_m(\sum_{j=0}^{n} a_{jm} x_j)$$

$$\frac{\partial L}{\partial b_m} = \frac{dL}{dr} \cdot \frac{\partial r}{\partial b_m} = \left(y_i - \overset{\frown}{F}(x_i)\right) \cdot \frac{d\left(-\overset{\frown}{F}(x_i)\right)}{db_m} = \left(\overset{\frown}{F}(x_i) - y_i\right) \cdot S_m\left(\sum_{j=0}^{n} a_{jm} x_j\right)$$

$$\frac{\partial L}{\partial b_m} = \text{"}error\text{"} \cdot \text{"}output\ of\ the\ m'th\ internal\ node\text{"}$$

Derivative to compute partial derivatives

For single layer network and for single observation we have

$$r_i = y_i - \tilde{F}(x_i)$$

$$L = \frac{1}{2}\left(y_i - \tilde{F}(x_i)\right)^2 = \frac{1}{2} r^2$$

$$\frac{dL}{dr} = \frac{2}{2} r = r = (y_i - \tilde{F}(x_i))$$

$$\tilde{F}(x; a, b) = \sum_{m=0}^{M} b_m \cdot S_m\left(\sum_{j'=0}^{n} a_{j'm} x_{j'}\right)$$

$$\frac{\partial L}{\partial a_{jm}} = \frac{dL}{dr}\frac{\partial r}{\partial a_{jm}} = (y_i - \tilde{F}(x_i))\frac{\partial\left(y_i - \tilde{F}(x_i)\right)}{\partial a_{jm}} = \left(\tilde{F}(x_i) - y_i\right) b_m \cdot \frac{\partial\left(S_m\left(\sum_{j'=0}^{n} a_{j'm} x_{j'}\right)\right)}{\partial a_{jm}}$$

$$\frac{\partial L}{\partial a_{jm}} = (\tilde{F}(x_i) - y_i) b_m \cdot S\left(\sum_{j'=0}^{n} a_{j'm} x_{j'}\right)\left(1 - S\left(\sum_{j'=0}^{n} a_{j'm} x_{j'}\right)\right) \cdot x_j$$

Our Prediction Schema

$$\overbrace{F}(x; a, b) = \sum_{m=0}^{M} b_m \cdot S_m\left(\sum_{j=0}^{n} a_{jm} x_j\right)$$

1. $S_0(z) = 1$ always
2. $S(z)' = S(z)(1 - S(z))$

Remarks about computation

1. Parallel computation is possible for all nodes in one layer . They all are completely independent during forward phase (and backward phase too)

2. If accuracy need not be very high then we can try instead of compute $S_j^{(l)}$ compute it's more easy surrogate

3. Computation is deterministic

What we evaluated so far?

In Lecture-3 we defined

$$\overbrace{S(\boldsymbol{a})} = \frac{1}{N}\sum_{i=1}^{N} L(y_i, F(x_i; \boldsymbol{a})) + \lambda P(a)$$

So far we consider case when:

- $L(y_i, \widehat{y_i}) = \frac{1}{2}(y_i - \widehat{y_i})$
- $P(a) = 0$

We evaluated partial derivatives for one examples:

$$\frac{1}{2}\left(y_i - \overbrace{F}(x_i)\right)^2$$

We can take all partial derivatives and write them into long vector

$$\nabla_a \overbrace{S(\boldsymbol{a})} = \nabla_a L(y_i, F(x_i; \boldsymbol{a}))$$

What about complete gradient?

- $\nabla_a \left(\frac{1}{N} \sum_{i=1}^{N} L(y_i, F(x_i; \boldsymbol{a})) \right) =$
  $\frac{1}{N} \cdot \nabla_a \left(\sum_{i=1}^{N} L(y_i, F(x_i; \boldsymbol{a})) \right) =$
  $\frac{1}{N} \cdot \left(\sum_{i=1}^{N} \nabla_a L(y_i, F(x_i; \boldsymbol{a})) \right)$

- This is real complete gradient

What is mini-batch

- Lets’ evaluate gradient not for complete loss function, which is objective to minimize
- Evaluate it via compute it for each observation
- Or for some subset.
  In this case it’s **mini-batch**
- Whole pass over the complete available data is called **“epoc”**. Does not matter what algorithm we use

Variants of gradient descend algorithms

- Batched steepest-descend
  In batched mode we include all train data.

- Online steepest-descend
  Take one train example at each time.
  Minuses: Depend of order of observation. Estimation of gradient based on one observation is not so reach.

- Iterative online steepest-descend with momentum

Store previous weights(parameters) which we're finding. And setup **gradient** to zero.

Evaluate gradient , but which contains one train example at each time.

Replace **gradient** with convex combination of evaluate gradient from previous step and current **gradient**

$$g = a \cdot (g_i) + (1-a) \cdot g \ , 0 \leq a \leq 1$$