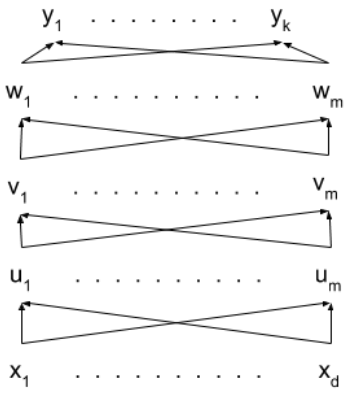
E0 306: Deep Learning: Theory and Practice

Lecturer: Navin Goyal
Scribe: Apoorv Saxena

Lecture #6
January 24, 2019

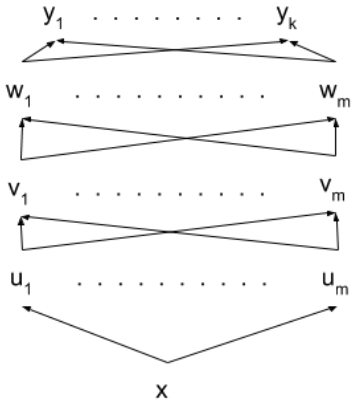
1 Review of neural networks



Claim: We can compute Jacobian $[\frac{dy_i}{dx_j}]$, $i\epsilon[k], j\epsilon[d]$ in $O(min(k,d) * (V+E))$ time.

Special cases:

(1) If $d = 1$, then we can compute in time $O(E)$ for any k



Note: *Automatic Differentiation*: For a given computer program P that calculates a function f, we can calculate the derivative of f in constant time ie. the same time it takes to evaluate the function.

2 modes: Forward mode AD and Backward mode AD (*Backpropagation*). Both touch each

Proof for forward mode AD:

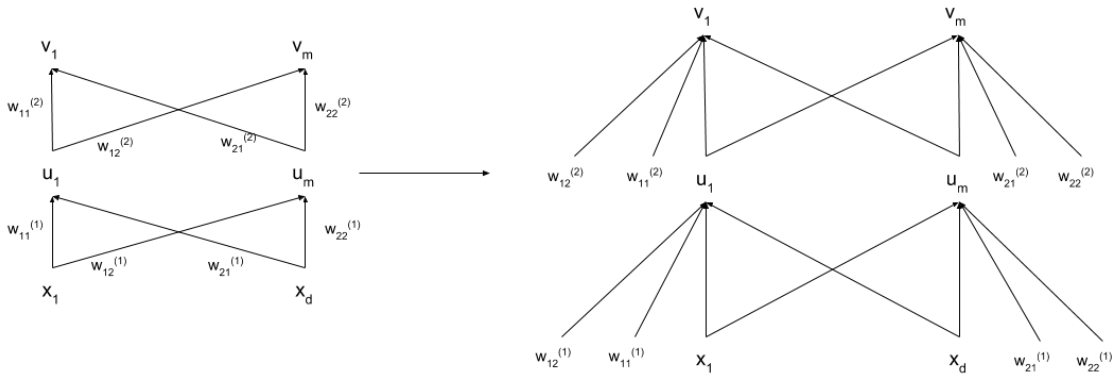
1. Compute $u_1(x), u_2(x), ..., u_m(x)$ and their derivatives $\frac{du_i}{dx}$, $i\epsilon[m]$
2. Compute $v_i(u_j)$ and their derivatives $\frac{dv_i}{du_j}$, $i, j\epsilon[m]$
3. Sum m^2 values to get $\frac{dv_i}{dx}$

4. Continue till last layer. No. of operations = 1 for each edge.

(2) If $k = d = m$ then time complexity is $O(l * m^3)$. Using fast matrix multiplication, it becomes $O(l * m^{2.39})$ approx.

(3) If we can solve this problem in $O(l*m^2)$ time then we can multiply matrics in $O(m^2)$ (non-trivial proof omitted)

1.1 Neural networks as computational graphs



This figure shows how a neural network can be unfolded into a computational graph as described earlier.

1.2 Initialization of weights for gradient descent

We assume that data is normalized. To do this set

$$x_i \leftarrow x_i - \frac{1}{m}\sum_j x_j$$

1.2.1 He initialization for ReLU networks [1]

Given a fully connected network of L layers where the size of each layer l is d_l with weight matrix $w^{(l)}$, we initialize each weight in $w^{(l)}$ as iid

$$w_{ij}^{(l)} \sim N(0, \sqrt{\frac{2}{d_l}})$$

(Note: This is similar to what was done for tanh networks in Xavier initialization by Glorot, Bengio 2010 where weights were initialized as:

$$w_{ij}^{(l)} \sim N(0, \sqrt{\frac{1}{d_l}})$$

1.2.2 Exploding and vanishing gradients

$\Delta_w L(w; S)$can become very large or very small.

Let there be a network with 1 scalar input and identity activation function.

$$y = w_L * w_{L-1} * ... * w_2 * w_1 * x$$

The i^{th} component of $\Delta_w y$ will be

$$\Delta_w y^i = w_L * w_{L-1} * ... * w_{i-1} * w_{i+1} * ... * w_2 * w_1 * x$$

if w_j are initialized as 2, this value explodes to become 2^{L-1} . On the other hand, if w_j are initialized as $\frac{1}{2}$, it becomes $2^{-(L-1)}$

However, if w_j are close to 1, the gradient neither explodes nor vanishes.

1.2.3 Proof of heuristic used in He initialization

Notation:

$$y^{(l)} = W^{(l)} x^{(l)} + b^{(l)}, 0 \leq l \leq L$$
$$x^{(l+1)} = ReLU(y^{(l)})$$
$$x^{(0)} = x$$

Assumptions:

- $x^{(l)}$ is a random variable with iid components
- $x^{(l)}$ and $w^{(l)}$ are independent

We will show the theoretical basis for He initialization.

$$Var(y_1^{(l)}) = Var(\sum_i W_{1i}^{(l)} x_1^{(l)})$$
$$= d_l Var(W_1 i^{(l)} x_1^{(l)})$$
$$= d_l Var(W_1 i^{(l)}) * \mathrm{E}[(x_i^{(l)})^2]$$

We will assume $b^{(l)} = 0$ in He initialization

$$\mathrm{E}(x_1^{(l)})^2 = \mathrm{E}[ReLU(y_1^{(l-1)})]^2$$
$$= \frac{1}{2}\mathrm{E}(y_1^{(l-1)})^2$$
$$= \frac{1}{2} Var(y_1^{(l-1)})$$

So we get

$$Var(y_1^{(l)}) = \frac{d_l}{2} Var(W_1 i^{(l)}) * Var(y_1^{(l-1)})$$

By recursively substituting for $Var(y^{(l)})$ we get

$$Var(y_1^{(L)}) = Var(y_1^{(1)}) * [\frac{d_2}{2} Var(W_1 i^{(1)})] * ... * [\frac{d_L}{2} Var(W_1 i^{(L)})]$$

We want each of these terms to be close to 1 so that the overall variance doesn't blow up or vanish. This can be achieved by setting

$$Var(W_{1i}^{(l)}) = \frac{2}{d_l}$$

2 Limitations of gradient based methods for deep learning

We work in the realizable setting of PAC learning. $\mathcal{X}$ is domain set, $\mathcal{Y}$ is label set and $\mathcal{H}$ is the set of hypotheses.

$$\mathcal{D}_x \sim \textit{distribution on } \mathcal{X}$$

$$(x, y) = (x, h(x))$$

for some unknown target $h \epsilon \mathcal{H}$

2.1 Gradient based methods

We define a loss function

$$L(w; S) = \sum_{(x,y)\epsilon S} L(w; (x, y))$$

We need to minimize $L(w; S)$ wrt w. For gradient descent, this is done by applying the following update.

$$w^{(t+1)} = w^{(t)} - \eta^{(t)} \Delta_w L(W; S)$$

From paper "Failures of gradient based deep learning (Schwartz et al., 2017)"

Claim 1: *If we choose hypothesis $h \epsilon \mathcal{H}$ uniformly, then with high probability, $\Delta_w L(w; S)$ is essentially independent of h*

Say

$$L(w) = E_{(x,y)\sim D} L(w; (x, y))$$

and $h \epsilon \mathcal{H}$ is the target function. $w \epsilon R^n$

Define $F_h(w)$ as

$$F_h(w) := L(w)$$

$$F_h(w) = E_{x \sim D} l(p_w(x), h(x))$$

Where p is the predictor parameterized by w and l is a loss function of type

$$l = \frac{1}{2}(y - \hat{y})^2$$

or

$$l = \gamma(y.\hat{y})$$

where γ is a 1-Lipschitz function.

Assumption: $F_h(W)$ is differentiable wrt W.

Define

$$Var(F, \mathcal{H}, w) := E_{h \epsilon_U \mathcal{H}} ||\Delta_w F_h(w) - E_{h' \epsilon_U \mathcal{H}} \Delta_w F_{h'}(w)||^2$$

If variance is small, then Claim-1 is true.

Theorem 1: Suppose that

1. *Functions in $\mathcal{H}$ are of type $\mathcal{X} \to \{-1, 1\}$*
2. *$E_{x \sim D_x} h(x) h'(x) = 0$ for distinct $h, h' \epsilon \mathcal{H}$*

3. $p_w(x)$ *is differentiable wrt* w *and satisfies*

$$E_x||\Delta_w p_w(x)||^2 \leq G(w)^2$$

for some function $G(w)$

4. *The loss* l *is either square loss or* $\gamma(y.\hat{y})$ *where* γ *is 1-Lipschitz*

Then

$$Var(\mathcal{H}, F, w) \leq \frac{G(w)^2}{|\mathcal{H}|}$$

Let $X = \{0,1\}^d$, $D_x :=$ uniform on $\{0,1\}^d$

$$\mathcal{H} := \{x \to (-1)^{<x,v>} | v \epsilon \{0,1\}^d\}$$

$\mathcal{H}$ essentially tells parity of x

$|\mathcal{H}| = 2^d$,

Fix and $v, v' \epsilon \{0,1\}^d$. Then

$$E_{x \sim D_x} h(x)h'(x) = E_{x \sim D_x}(-1)^{<v,x>}(-1)^{<v',x>}$$

$$= E_{x \sim D_x}(-1)^{<v+v',x>}$$

$$= 0 \; if \; v \neq v' \; and \; 1 \; otherwise$$

let $vec(h) = v$ such that $h(x) := x \to (-1)^{<x,v>}$

So

$$\{vec(h) : h \epsilon \mathcal{H}\} \; \epsilon \; R^{2^d}$$

is an orthogonal basis for R^{2^d}

(proof continued in next lecture)

Remark: (graph plot showing accuracy as a function of d)

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ICCV '15, pages 1026–1034, Washington, DC, USA, 2015. IEEE Computer Society.