6.S897 Algebra and Computation April 23, 2012

Lecture 19 - Computing Partial Derivates and Depth Reduction

Instructor: Madhu Sudan *Scribe: Travis Hance*

1 Overview

In this lecture we cover:

- [1] If we can compute polynomial f with an arithmetic circuit of size s, then we can compute all of its partial derivatives $(\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_n})$ in size $O(s)$.
- [3] Depth reduction: If f can be computed with a circuit of size s, then f can be computed in size $\mathsf{poly}(s)$ and depth $O(\log^2 s)$.

2 Clarification of arithmetic circuit model

We might ask the question: how universal are arithmetic circuits for universal computation? We cannot compute everything using arithmetic circuits; for instance, we cannot compute square roots or compute the gcd of two polynomials, because whatever we compute must be a polynomial in the inputs.

However, we may wonder about the something such as the determinant: it is a polynomial in its input variables, and there is a polynomial time algorithm (Gaussian elimination) to compute it. However, this algorithm involves branching: we have to test whether a value is nonzero in order to see if it can be used as a pivot. However, this is not an issue: if we just make a circuit to compute the determinant treating input variables as formal variables, then we do not need to use branching. In the end we will have a circuit computing f/g which equals the determinant wherever it is defined. We already know from the last lecture that we can eliminate division and get a circuit for computing determinant.

This method works in general for computing polynomials with arithmetic circuits if you can compute them with branches of the form "does $y_i = 0$?".

Also note that there are known explicit curcuits for computing the determinant by inductively computing the characteristic polynomial [2].

3 Computing partial derivates [1]

The key idea is to use a "top-down" approach rather than a "bottom-up" approach. That is, instead of computing, for each gate, the partial derivatives of that gate with respect to the input variables, we compute the partial derivatives of the output with respect to gates.

Suppose that a polynomial $f(x_1, \ldots, x_n)$ is computed by a circuit of size s, that is, a series of s steps of the form $y_i = y_j \lozenge y_k$ (or $y_i = y_j \lozenge x_k$ or $y_i = x_j \lozenge x_k$) where $\lozenge \in \{+, \times\}$. Rather than thinking of the end result y_s as a function of the input variables $x_1, \ldots, x_n$, we imagine a function $g(x_1, \ldots, x_n, y_1, \ldots, y_s)$ which computes the end result that we can write in many different ways. More specifically,

- Let $g_s(x_1, \ldots, x_n, y_1, \ldots, y_s) = y_s$
- For $1 \leq i \leq s$, let $g_{i-1} = g_i|_{y_i \leftarrow y_j \Diamond y_k}$, that is, g_i is obtained by substituting $y_j \Diamond y_k$ for y_i in g_{i+1}, in the circuit, y_i is computed as $y_j \Diamond y_k$. (By abuse of notation, we will always write y_j and y_k when in fact either could be one of the inputs $x_1, \ldots, x_n$ instead.)

Thus for any i, g_i is a polynomial in $x_1, \ldots, x_n, y_1, \ldots, y_i$, and so g_0 is the original polynomial f in only the input variables. Thus our goal is to compute

$$\left(\frac{\partial g_0}{\partial x_1}, \ldots, \frac{\partial g_0}{\partial x_n}\right)$$

The idea is to compute $\{\frac{\partial g_i}{\partial y_j}(x_1, \ldots, x_n, y_1, \ldots, y_s)\}$ for all i, j. Note that while there are clearly more than $O(s)$ such derivatives, many of them will be equal and so we will only need $O(s)$ gates.

Since $g_s(x_1, \ldots, x_n, y_1, \ldots, y_n) = y_s$, we initially we have $\partial g_s/\partial y_s = 1$, and $\frac{\partial g_s}{\partial y_j} = 0$ for all $j < s$.

Now consider g_{i-1} for $i \leq s$. Recall that $g_{i-1} = g_i|_{y_i \leftarrow y_j \Diamond y_k}$. Naturally, we have

$$\frac{\partial g_{i-1}}{\partial y_i} = 0 \tag{1}$$

since y_i does not appear anywhere in the polynomial g_{i-1}. Since we are just making a substitution $y_i \leftarrow y_j \Diamond y_k$, by the chain rule,

$$\frac{\partial g_{i-1}}{\partial y_j} = \frac{\partial g_i}{\partial y_j} + \frac{\partial g_i}{\partial y_i} \cdot \left(\frac{\partial}{\partial y_j}(y_j \Diamond y_k)\right) \tag{2}$$

where of course $(\partial/\partial y_j)(y_j + y_k) = 1$ and $(\partial/\partial y_j)(y_j \times y_k) = y_k$. We get a similar equation for $\partial g_{i-1}/\partial y_k$. Finally, for all other y_ℓ ($\ell \neq i, j, k$) we get

$$\frac{\partial g_{i-1}}{\partial y_\ell} = \frac{\partial g_i}{\partial y_\ell} \tag{3}$$

Remember that in identities (2) and (3), the equality is not of formal polynomials in $x_1, \ldots, x_n, y_1, \ldots, y_s$ but $y_1, \ldots, y_s$ should be considered as polynomials in $x_1, \ldots, x_n$.

Thus, using (1), (2), and (3) we can compute all the derivatives $\partial g_{i-1}/\partial y_j$ for all j from the partial derivates $\partial g_i/\partial y_j$, using only a constant number of gates. Thus in $O(s)$ gates we can compute the partial derivates $\partial g_0/\partial x_j$ as desired.

4 Depth Reduction [3]

In this section we show the following result:

Theorem 4.1. *Let ϕ be a circuit computing a polynomial f. If the size of ϕ is s and $\deg f \leq s$ then there is a circuit computing f with size $\mathsf{poly}(s)$ and depth $O(\log^2(s))$.*

From the last lecture we know that we there are small arithmetic circuits computing the homogeneous parts of f. Thus we assume that f is homogeneous and that ϕ is a homogeneous circuit, i.e., the output of each gate is a homogeneous polynomial.

Now some notation: for any gate v, let f_v be the polynomial in $x_1, \ldots, x_n$ computed at gate v. Furthermore, if v and w are gates, we can think of v as computing a polynomial from inputs $x_1, \ldots, x_n, w$ by "cutting off" the gate w and pretending its output is just another input to the circuit. Thus, we can write $f_v(x_1, \ldots, x_n) = g_{v,w}(x_1, \ldots, x_n, w)|_{w=f_w}$. We then define

$$\partial_w f_v = \left.\frac{\partial g_{v,w}}{\partial w}\right|_{w=f_w} \tag{4}$$

This is the notion of a partial derivative of a gate with respect to another gate that we will use in this construction.

To compute f using polylogarithmic depth, we will proceed in stages. At stage i we will compute:

- Compute all polynomials f_v such that $2^i < \deg f_v \leq 2^{i+1}$.
- Compute all $\partial_w f_v$ such that $2^i < \deg v - \deg w \leq 2^{i+1}$ and $\deg w \leq \deg v \leq 2 \deg w$.

Each stage will be done with $\mathsf{poly}(s)$ size and with a depth-2 (unbounded fan-in) circuit. Any unbounded fan-in gate can we replaced with a constant fan-in, logarithmic depth circuit.

To compute all these things, we rely on the following:

Lemma 4.2. *Let*

$$\mathcal{G}_m \stackrel{\text{def}}{=} \{t : t = t_1 \times t_2, \deg(t) > m, \deg(t_1), \deg(t_2) \leq m\}. \tag{5}$$

Then,

(i) For all gates v with $m < \deg(f_v) \leq 2m$, we have

$$f_v = \sum_{t \in \mathcal{G}_m} f_t \partial_t f_v \tag{6}$$

(ii) For all gates v, w with $\deg(w) \leq m < \deg(v) \leq 2\deg(w)$, we have

$$\partial_w f_v = \sum_{t \in \mathcal{G}_m} (\partial w f_t)(\partial_t f_v) \tag{7}$$

Proof. (i) We use induction on the depth of v. The base case has $v \in \mathcal{G}_m$. For this, we want to show that

$$f_v = f_v \cdot \partial_v f_v + \sum_{t \in \mathcal{G}_m \setminus \{v\}} f_t \cdot \partial_t f_v \tag{8}$$

but this is true because $\partial_v f_v = 1$, and since there can be no path in ϕ from t to v for any other t, we must have $\partial_t f_v = 0$.

For induction, suppose $v = v_1 \Diamond v_2$, where we already know the statement (6) holds for v_1 and v_2. In the '+' case, we have $f_v = f_{v_1} + f_{v_2}$ and so

$$\begin{aligned} f_v &= \sum_{t \in \mathcal{G}_m} f_t \partial_t f_{v_1} + \sum_{t \in \mathcal{G}_m} f_t \partial_t f_{v_2} \\ &= \sum_{t \in \mathcal{G}_m} f_t (\partial_t f_{v_1} + \partial_t f_{v_2}) \\ &= \sum_{t \in \mathcal{G}_m} f_t (\partial_t f_v) \end{aligned}$$

For the '×' case, assume without loss of generality that $\deg f_{v_1} \geq \deg f_{v_2}$. Then we must have $\deg(v_2) \leq m$ and so $\partial_t f_{v_2} = 0$. Thus, by the product rule,

$$\partial_t f_v = (\partial_t f_{v_1})(f_{v_2}) + (\partial_t f_{v_2})(f_{v_1}) = (\partial_t f_{v_1})(f_{v_2}). \tag{9}$$

Hence, $f_v = f_{v_1} f_{v_2}$ so

$$\begin{aligned} f_v &= \left(\sum_{t \in \mathcal{G}_m} f_t \partial_t f_{v_1} \right) f_{v_2} \\ &= \sum_{t \in \mathcal{G}_m} f_t \cdot (f_{v_2} \partial_t f_{v_1}) \\ &= \sum_{t \in \mathcal{G}_m} f_t \partial_t f_v \end{aligned}$$

which completes the induction step.

(ii) We simply take the derivative of the result from (i). Expanding with the product rule gives

$$\partial_w f_v = \sum_{t \in \mathcal{G}_m} ((\partial_w f_t)(\partial_t f_v) + (f_t)(\partial_w \partial_t f_v)) \tag{10}$$

But $\partial_w \partial_t f_v$ is always 0 since $\deg(v) \leq 2 \deg(w) < \deg(w) + \deg(t)$. Thus this simplifies to the desired result.

□

Now, Lemma (4.2) immediately lets us complete the construction outlined above, thus proving Theorem (4.1).

References

[1] W. Baur, V. Strassen. *The complexity of partial derivatives.* Theoretical Computer Science, Volume 22, Issue 3, February 1983, Pages 317-330, ISSN 0304-3975, 10.1016/0304-3975(83)90110-X.

[2] S. J. Berkowitz. *On computing the determinant in small parallel time using a small number of processors.* Inf. Prod. Letters 18, pages 147-150, 1984.

[3] L. G. Valiant, S. Skyum, S. Berkowitz, C. Rackoff. *Fast parallel computation of polynomials using few processors.* SIAM J. Comput. 12(4), pp. 641-644, 1983.