Chebyshev Polynomials and Approximation Theory in Theoretical Computer Science and Algorithm Design

(Talk for MIT's Danny Lewin Theory Student Retreat, 2015)

Cameron Musco

October 8, 2015

Abstract

I will talk about low degree polynomials that are small on the interval [0,1] but jump up very rapidly outside of that interval. First I'll try to convince you why you should care about such polynomials, which have a ton of applications in complexity, optimization, learning theory, and numerical linear algebra. In particular, I will introduce the Chebyshev polynomials and then try to explain the 'magic' behind them which makes them so useful in the above applications.

A very nice reference if you're interested in this area is *Faster Algorithms via Approximation Theory* by Sachdeva and Vishnoi. Some of my proofs are taken from that book. I also pulled my quantum proof from a talk by Scott Aaronson on the polynomial method in TCS: `www.scottaaronson.com/talks/polymeth.ppt`

1 Step Polynomials and Their Applications

We will refer to a 'jump polynomial' as a degree d polynomial $p(x)$ such that, for $x \in [0,1]$, $|p(x)| \leq \epsilon$. For $x > 1+\gamma$, $p(x) > 1$.

It is not hard to see that such polynomials exist for $d = O\left(\log(1/\epsilon)/\gamma\right)$. Just set $p(x) = \frac{1}{(1+\gamma)^d}x^d$. Clearly $p(x) \geq 1$ for $x \geq 1+\gamma$. Additionally, of $x \in [0,1]$ we have:

$$p(x) \leq \frac{1}{(1+\gamma)^d} \leq \frac{1}{\left((1+\gamma)^{1/\gamma}\right)^{\log(1/\epsilon)}} \leq \frac{1}{e^{\log(1/\epsilon)}} \leq \epsilon.$$

However we can do much better. In fact using *Chebyshev Polynomials*, degree $d = O\left(\log(1/\epsilon)/\sqrt{\gamma}\right)$ suffices. The fact that you can achieve this improvement should be surprising to you. Additionally, it is optimal.

Before we talk at all about Chebyshev polynomials let me give some applications of this fact. The two most important applications (from my unbiased viewpoint) are eigenvector computation/PCA and linear systems solving/convex optimization. However, I am talking to the theory group so let me try two other options first:

Learning Theory

A polynomial threshold function of degree d is just some function $h(x) = sign\,(p(x))$ where $p(x)$ is a degree d *multivariate* polynomial. The most basic version, when $d = 1$ is a linear threshold function, aka a halfspace, aka a linear classifier. Learning these functions is one of the most (or the most) studied problems in learning theory. Its what perceptron does, what SVM does, what boosting and neural networks were originally used for etc.

You can provably learn a halfspace in polynomial time using linear programming. By building on these techniques, you can learn a degree d polynomial threshold function over n variables in $n^{O(d)}$ time.

Hence a number of results in learning theory attempt to show that certain functions are well approximated by low degree polynomial threshold functions and so can be learned efficiently. One good example is *Learning Intersections and Thresholds of Halfspaces* by Klivans, O'Donnell, and Servedio [FOCS '02].

Here I will talk about a result in STOC '01 Klivans and Servedio showing how to learn DNF functions over n variables with s clauses in time $2^{O(n^{1/3}\log s)}$. This solved a longstanding open question, matching a 1968 lower bound of Minsky and Papert and beating the previous algorithmic result of $2^{O(n^{1/2}\log n \log s)}$.

Really they show that a DNF with n variables and s clauses can be approximated by a polynomial of degree $n^{1/3}\log s$, which they did by showing first that a DNF with s terms, where each conjunction has only k variables in it can be approximated by a polynomial of degree $O(\sqrt{k}\log s)$. Which they did using Chebyshev polynomials. How? Its ridiculously simple.

Consider evaluating a single term $x_1 \wedge x_2 \wedge ... \wedge x_k$. We want a threshold that is high if all terms are true and low otherwise. Set our threshold to be:

$$f(x_1, x_2, ..., x_k) = T_{\sqrt{k}\log S}\left(\frac{x_1 + x_2 + ... + x_k}{k}\right)$$

Note that since $T_{\sqrt{k}\log S}$ is a degree $\sqrt{k}\log s$ polynomial, this is a degree $\sqrt{k}\log s$ multivariate polynomial.

If the clause is true then we have $\frac{x_1+x_2+...x_k}{k} = 1$. If not true, we have $\frac{x_1+x_2+...x_k}{k} \leq 1 - \frac{1}{k}$. Therefore, by the jump property of $T_{\sqrt{k}\log S}$, we know that $f(x_1, x_2, ...x_k) \leq \frac{1}{2S}$ if the clause isn't true and $f(x_1, x_2, ...x_k) \geq 1$ otherwise.

Now if we just add all these functions together, we can evaluate the DNF by returning true if the sum ≥ 1 and false otherwise. If at least 1 term is true the sum will be ≥ 1. If all terms are false, the sum will be $\leq s \cdot \frac{1}{2s} \leq \frac{1}{2}$. Note this is NOT a probabilistic thing. This polynomial is just a way to evaluate the DNF exactly. Hence, if we learn the polynomial, we learn the DNF.

I won't go into details but note that without Chebyshev acceleration they just get $2^{O(n^{1/2}\log s)}$ so are far from the lower bound and don't beat previous results based on other methods.

Quantum Complexity Theory

Above we saw an algorithm based off Chebyshev polynomials. However, the optimality of these polynomials is also often used for lower bounds. This is especially popular in quantum complexity theory, where the general technique is referred to as the *polynomial method*.

There is a famous algorithm in quantum complexity called 'Grover's search algorithm' that allows you to find a non zero value in a vector $x \in [0,1]^n$ using $O(\sqrt{n})$ quantum queries. I know,

quantum is pretty messed up. You can use the optimality of Chebyshev polynomials to show that this is tight.

First off, note that Grover's search solves $OR(x)$ in $O(\sqrt{n})$ queries to x, since if there is at least one non zero value in x, it returns its index, indicating that $OR(x) = 1$.

The number of quantum queries required to compute a boolean function like $OR(x)$ (denote it as $Q(OR)$) is lower bounded by what quantum people refer to as the approximate degree of that function $\widetilde{deg}(OR)$. This is just the minimum degree of a (multivariate) polynomial $p(x_1, ...x_n)$ that approximates that function up to uniform error say $1/3$ (i.e. differs on any input by at most $1/3$ from the OR of that input). Note that $OR(x)$ only outputs either 0 or 1. $p(x)$ outputs general real numbers.

The bound that $Q(OR) \geq \widetilde{deg}(OR)$ is essentially because the probability that a quantum algorithm outputs 1 on a given input can be written as a polynomial with degree equal to the number of queries made to the input. So to be right with probability $2/3$ on any input the algorithm's success probability better be a polynomial with degree $Q(OR)$ that is a $1/3$ uniform approximation to $OR(x)$.

Now we don't want to work with multivariate polynomials. But we can use *symmetrization*, defining the univariate polynomial:

$$h(k) = \mathop{\mathbb{E}}_{\|x\|_1 = k} p(x).$$

You can show that the degree of $h(\cdot)$ is upper bounded by the degree of $p(\cdot)$, although I will not discuss that here. So lower bounding $Q(OR)$ just requires lower bounding $deg(h)$.

Now consider $p(x)$ approximating $OR(x)$, which is computed by Grover search. If $p(x)$ approximates OR, then we have $h(0) \approx 0$ and for $x = 1, ...n$, $h(x) \approx 1$. Flipping this polynomial horizontally and vertically and scaling the x axis by $1/n$, we have a polynomial that is small on $[0, 1]$ and jumps to near 1 at $1 + 1/n$. By the optimality of the Chebyshev polynomials this must have degree $\Omega(\sqrt{n})$ giving us the lower bound for Grover search.

Linear System Solvers

Chebyshev polynomials place a very important role in accelerated methods for convex optimization (linear system solvers as a special case). The methods are widely studied by numerical computation people, theorists, machine learning people, etc.

Simple iterative linear system solvers or convex optimization routines like Gradient Descent, Richardson Iteration run in time $O(\text{nnz}(\mathbf{A}) \log(n/\epsilon)/\kappa)$, where κ is the condition number of the matrix (or the strong convexity parameter in convex optimization). This can be improved to $O(\text{nnz}(\mathbf{A}) \log(n/\epsilon)/\sqrt{\kappa})$ using accelerated solvers like conjugate gradient, Nesterov's accelerated gradient descent, and Chebyshev iteration.

To solve $\mathbf{Ax} = \mathbf{b}$ we want to return $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$, but computing the inverse exactly is too slow. Any iterative method is just applying a polynomial of $\mathbf{A}$ to $\mathbf{b}$ whose degree is equal to the number of iterations in the method. That is, we are computing $\tilde{\mathbf{x}} = p(\mathbf{A})\mathbf{b}$ as an approximation to $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$.

Say we write $\tilde{\mathbf{x}}$ as $p(\mathbf{A})\mathbf{Ax}$. We can decompose into the eigenbasis to get:

$$\mathbf{V}p(\mathbf{\Lambda})\mathbf{V}^\top\mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top = \mathbf{V}h(\mathbf{\Lambda})\mathbf{V}^\top.$$

where $h(x) = x \cdot p(x)$. Now h acts on the eigenvalues of $\mathbf{A}$, which fall in the range $[\lambda_n, \lambda_1]$. We want h to to be near one on all these eigenvalues – so that $\mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top$ is close to the identity. However,

clearly $h(0) = 0 \cdot p(0) = 0$. So...its just like the function we were looking at for Grover's search. Its zero at 0 and jumps to one on $[\lambda_n, \lambda_1]$. Flipping and scaling, we see that it is a jump polynomial with gap $\gamma = \lambda_1/\lambda_n = 1/\kappa$.

Eigenvector Computation

Let $gap = (\lambda_1 - \lambda_2)/\lambda_2$ be the eigengap separating the top eigenvalue of a matrix from its second eigenvalue. The power method converges in $O(\log(1/\epsilon)/gap)$ iterations. Lanczos or Chebyshev iteration use Chebyshev polynomials to get $O(\log(1/\epsilon)/\sqrt{gap})$. I'm not going to explain this one in detail – it is a direct application of jump polynomials, where we scale and shift such that λ_2 goes to 1 and λ_1 goes to $1 + gap$.

Other Applications

- Approximating other matrix functions such as $exp(\mathbf{M})$, random walks, etc...
- Streaming entropy approximation [Harvey, Nelson, Onak FOCS '08]
- Quadrature rules.
- *Lower bounds in communication complexity and learning theory via analytic methods*, Alexander Sherstov's PhD thesis 2009.
- I even found a paper in cryptography using them to replace monomials in the Diffie-Hellman and RSA algorithms [Fee and Monagan. *Cryptography using Chebyshev polynomials* 2004.]

2 Optimality of Chebyshev Polynomials

> *There's only one bullet in the gun. It's called the Chebyshev polynomial.* – Rocco Servedio via Moritz Hardt (*Zen of Gradient Descent* blog post).

It turns out, that the optimal jump polynomials are given by the Chebyshev polynomials (of the first kind). These are a family of polynomials, $T_d(\cdot)$ for all degrees $d \geq 0$.

The extremal properties of Chebyshev polynomials were first proven by the Markov brothers, both who were Chebyshev's students. Specifically, they showed that the Chebyshev polynomials achieve the maximum derivative (second derivative, third derivative, etc.) on $[-1, 1]$ as a ratio of max value on $[-1, 1]$. One brother died soon after of tuberculousis at the age of 25. The other is the Markov we all know about.

We're going to show something a bit different that is more directly related to the jump property that we care about.

Theorem 1 (Optimality of Chebyshev Polynomials)**.** *For any polynomial $g(\cdot)$ of degree d with $-1 \leq g(x) \leq 1$ for $x \in [-1, 1]$, letting $T_d(x)$ be the degree d Chebyshev polynomial, we have for any $x \notin [-1, 1]$:*

$$|g(x)| \leq |T_d(x)|.$$

Further, $-1 \leq T(x) \leq 1$

In words, this theorem says that T_d grows faster outside of $[-1, 1]$ than *any other polynomial* that is bounded in the range $[-1, 1]$.

Definition 2 (Recursive Definition of Chebyshev Polynomials)**.** *The Chebyshev Polynomials can be defined by the recurrence:*

- $T_0(x) = 1$
- $T_1(x) = x$
- $T_d(x) = 2xT_{d-1}(x) - T_{d-2}(x)$

This recurrence gives us the lemma

Lemma 3 (Cosine Intepretation of Chebyshev Polynomials)**.**

$$T_d(\cos\theta) = \cos(d\theta)$$

Proof. How? By induction. How else?

- $T_0(\cos\theta) = 1 = \cos(0 \cdot \theta)$.
- $T_1(\cos\theta) = \cos\theta = \cos(1 \cdot \theta)$.

And by induction:

$$\begin{aligned} T_d(\cos\theta) &= 2\cos\theta \cdot T_{d-1}(\cos\theta) - T_{d-2}(\cos\theta) \\ &= 2\cos\theta \cdot \cos((d-1)\theta) - \cos((d-2)\theta) \\ &= \cos(d\theta) \end{aligned}$$

The last inequality follows from doing a Google image search for 'cosine addition formula' and finding:

$$\cos a \cos b = \frac{\cos(a+b) + \cos(a-b)}{2}.$$

Plugging in $a = (d-1)\theta$ and $b = \theta$ gives:

$$\begin{aligned} \cos(a+b) &= 2\cos a \cos b - \cos(a-b) \\ \cos(d\theta) &= 2\cos\theta\cos((d-1)\theta) - \cos((d-2)\theta). \end{aligned}$$

Or you can just apply the normal cosine addition formula to $\cos((d-1)\theta + \theta)$ and $\cos((d-1)\theta - \theta)$. □

Well why does Lemma 3 matter? First, it immediately gives:

Corollary 4 (Boundedness Property of Chebyshev Polynomials)**.** *For any d and $x \in [-1, 1]$:*

$$-1 \leq T_d(x) \leq 1.$$

Proof. Let $\theta = \cos^{-1} x$. By Lemma 3, $T_d(x) = T_d(\cos(\theta)) = \cos(d\theta) \in [-1, 1]$ since cosine is always in $[-1, 1]$. □

Second, it gives:

Corollary 5 (Alternation Property of Chebyshev Polynomials)**.** *$T_d(x)$ alternates between -1 and 1 d times on $[-1, 1]$.*

.

Proof. For $x \in \{\cos(0), \cos(2\pi/d), \cos(4\pi/d), ..., \cos(d\pi/d)\}$ we have $T_d(x) = \cos(2i\pi) = 1$. For $x \in \{\cos(\pi/d), \cos(3\pi/d), \cos(5\pi/d), ...\}$ we have $T_d(x) = \cos((2i+1)\pi) = -1$. There are $d+1$ such values. So the function alternates between -1 and 1 d times. □

This alternation property is critical. it makes proving the optimality theorem super easy.

Proof of Theorem 1. Assume for contradiction that there exists some degree d polynomial $g(\cdot)$ with $-1 \leq g(x) \leq 1$ for $x \in [-1, 1]$ and some $y \notin [-1, 1]$ with $|g(y)| \geq |T_q(y)|$.

Let $h(x) = \frac{T_d(y)}{g(y)} \cdot g(x)$ Then $T_d(x) - h(x)$ is a degree d polynomial with a 0 at y. However, it also has d zeros in $[-1, 1]$ since every time T_d alternates between -1 and 1 it crosses $h(x)$ (which is bounded in $[-1, 1]$ since $\left|\frac{T_d(y)}{g(y)}\right| < 1$. Hence, $T_d(x) - h(x)$ has $d + 1$ zeros but only has degree d, a contradiction. That's it. □

3 Understanding the Optimal Growth Rate

It is not very hard to explicitly compute for Chebyshev polynomials that

$$T_d(1 + \gamma) \geq 2^{d\sqrt{\gamma}-1}.$$

Thus if we set $d = O\left(\frac{\log(1/\epsilon)}{\sqrt{\gamma}}\right)$, and scale appropriately, we have a polynomial such that $P(1 + \gamma) = 1$ at $1 + \gamma$ and $|p(x)| \leq \epsilon$ for $x \in [-1, 1]$.

We can also prove it inductively, starting with $T_1(\cdot)$ and using the recurrence relation. However this doesn't give us any intuition for what we can achieve this type of growth with a $1/\sqrt{\gamma}$ instead of a $1/\gamma$ dependence.

In their monograph, Vishnoi and Sachdeva discuss a proof based on the fact that a random walk of length d will deviate from the origin by at most $O(\sqrt{d})$ with good probability. This sort of gives a better intuition. There are also some intuitive understanding of how accelerated gradient descent methods (which achieve similar bounds to Chebyshev polynomials) work. However, to me, a large part of Chebyshev polynomials can still be attributed to 'magic'.