Chapter 1

Expectation and Quick Sort

By Sariel Har-Peled, December 10, 2019

Everybody knows that the dice are loaded
Everybody rolls with their fingers crossed
Everybody knows the war is over
Everybody knows the good guys lost
Everybody knows the fight was fixed
The poor stay poor, the rich get rich
That’s how it goes
Everybody knows

Everyday knows, Leonard Cohen

1.1. Basic probability

Here we recall some definitions about probability. The reader already familiar with these definition can happily skip this section.

1.1.1. Formal basic definitions: Sample space, \( \sigma \)-algebra, and probability

A sample space \( \Omega \) is a set of all possible outcomes of an experiment. We also have a set of events \( \mathcal{F} \), where every member of \( \mathcal{F} \) is a subset of \( \Omega \). Formally, we require that \( \mathcal{F} \) is a \( \sigma \)-algebra.

Definition 1.1.1. A single element of \( \Omega \) is an elementary event or an atomic event.

Definition 1.1.2. A set \( \mathcal{F} \) of subsets of \( \Omega \) is a \( \sigma \)-algebra if:

(i) \( \mathcal{F} \) is not empty,
(ii) if \( X \in \mathcal{F} \) then \( \overline{X} = (\Omega \setminus X) \in \mathcal{F} \), and
(iii) if \( X, Y \in \mathcal{F} \) then \( X \cup Y \in \mathcal{F} \).

More generally, we require that if \( X_i \in \mathcal{F} \), for \( i \in \mathbb{Z} \), then \( \cup_i X_i \in \mathcal{F} \). A member of \( \mathcal{F} \) is an event.

As a concrete example, if we are rolling a dice, then \( \Omega = \{1, 2, 3, 4, 5, 6\} \) and \( \mathcal{F} \) would be the power set of all possible subsets of \( \Omega \).

Definition 1.1.3. A probability measure \( \mathbb{P} : \mathcal{F} \to [0, 1] \) assigning probabilities to events. The function \( \mathbb{P} \) needs to have the following properties:

(i) ADDITIVE: for \( X, Y \in \mathcal{F} \) disjoint sets, we have that \( \mathbb{P}[X \cup Y] = \mathbb{P}[X] + \mathbb{P}[Y] \), and
(ii) \( \mathbb{P}[\Omega] = 1 \).

Definition 1.1.4. A probability space is a triple \( (\Omega, \mathcal{F}, \mathbb{P}) \), where \( \Omega \) is a sample space, \( \mathcal{F} \) is a \( \sigma \)-algebra defined over \( \Omega \), and \( \mathbb{P} \) is a probability measure.

Definition 1.1.5. A random variable \( f \) is a mapping from \( \Omega \) into some set \( \mathcal{G} \). We require that the probability of the random variable to take on any value in a given subset of values is well defined. Formally, for any subset \( U \subseteq \mathcal{G} \), we have that \( f^{-1}(U) \in \mathcal{F} \). That is, \( \mathbb{P}[f \in U] = \mathbb{P}[f^{-1}(U)] \) is defined.

\( ^{\text{\textsuperscript{\textcopyright}}}\)This work is licensed under the Creative Commons Attribution-Noncommercial 3.0 License. To view a copy of this license, visit http://creativecommons.org/licenses/by-nc/3.0/ or send a letter to Creative Commons, 171 Second Street, Suite 300, San Francisco, California, 94105, USA.
Going back to the dice example, the number on the top of the dice when we roll it is a random variable. Similarly, let \( X \) be one if the number rolled is larger than 3, and zero otherwise. Clearly \( X \) is a random variable.

We denote the probability of a random variable \( X \) to get the value \( x \), by \( \mathbb{P}[X = x] \) (or sometime \( \mathbb{P}[x] \), if we are lazy).

**1.2. Expectation and conditional probability**

**1.2.1. Expectation**

**Definition 1.2.1 (Expectation).** The expectation of a random variable \( X \), is its average. Formally, the expectation of \( X \) is

\[
\mathbb{E}[X] = \sum_{x} x \mathbb{P}[X = x].
\]

**Lemma 1.2.2 (Linearity of expectation).** Linearity of expectation is the property that for any two random variables \( X \) and \( Y \), we have that \( \mathbb{E}[X + Y] = \mathbb{E}[X] + \mathbb{E}[Y] \).

**Proof:**

\[
\mathbb{E}[X + Y] = \sum_{\omega \in \Omega} \mathbb{P}[\omega] (X(\omega) + Y(\omega)) = \sum_{\omega \in \Omega} \mathbb{P}[\omega] X(\omega) + \sum_{\omega \in \Omega} \mathbb{P}[\omega] Y(\omega) = \mathbb{E}[X] + \mathbb{E}[Y].
\]

**Example 1.2.3.** Let \( F \) be a boolean formula with \( n \) variables in CNF form, with \( m \) clauses, where each clause has exactly \( k \) literals. A random assignment for \( F \), where value 0 or 1 is picked with probability 1/2, satisfies in expectation \((1 - 2^{-k})m\) of the clauses.

**1.2.2. Conditional probability**

**Definition 1.2.4 (Conditional Probability).** The conditional probability of \( X \) given \( Y \), is the probability that \( X = x \) given that \( Y = y \). We denote this quantity by \( \mathbb{P}[X = x \mid Y = y] \).

One useful way to think about the conditional probability \( \mathbb{P}[X \mid Y] \) is as a function, between the given value of \( Y \) (i.e., \( y \)), and the probability of \( X \) (to be equal to \( x \)) in this case. Since in many cases \( x \) and \( y \) are omitted in the notation, it is somewhat confusing.

The conditional probability can be computed using the formula

\[
\mathbb{P}[X = x \mid Y = y] = \frac{\mathbb{P}[(X = x) \cap (Y = y)]}{\mathbb{P}[Y = y]}.
\]

For example, let us roll a dice and let \( X \) be the number we got. Let \( Y \) be the random variable that is true if the number we get is even. Then, we have that

\[
\mathbb{P}[X = 2 \mid Y = true] = \frac{1}{3}.
\]

**Definition 1.2.5.** Two random variables \( X \) and \( Y \) are independent if \( \mathbb{P}[X = x \mid Y = y] = \mathbb{P}[X = x] \), for all \( x \) and \( y \).
Observation 1.2.6. If $X$ and $Y$ are independent then $\Pr[X = x \mid Y = y] = \Pr[X = x]$ which is equivalent to $\frac{\Pr[X = x \cap Y = y]}{\Pr[Y = y]} = \Pr[X = x]$. That is, $X$ and $Y$ are independent, if for all $x$ and $y$, we have that

$$\Pr[X = x \cap Y = y] = \Pr[X = x] \Pr[Y = y].$$

Remark. Informally, and not quite correctly, one possible way to think about conditional probability $\Pr[X = x \mid Y = y]$ is as measuring the benefit of having more information. If we know that $Y = y$, do we have any change in the probability of $X = x$?

Lemma 1.2.8. If $X$ and $Y$ are two random independent variables, then $\mathbb{E}[XY] = \mathbb{E}[X] \mathbb{E}[Y]$.

1.2.3. Application: Approximating $k$-SAT

We remind the reader that an instance of 3SAT is a boolean formula, for example $F = (x_1 + x_2 + x_3)(x_4 + \overline{x_1} + x_2)$, and the decision problem is to decide if the formula has a satisfiable assignment. Interestingly, we can turn this into an optimization problem.

Max 3SAT

- **Instance**: A collection of clauses: $C_1, \ldots, C_m$.
- **Question**: Find the assignment to $x_1, \ldots, x_n$ that satisfies the maximum number of clauses.

Clearly, since 3SAT is NP-COMPLETE it implies that Max 3SAT is NP-HARD. In particular, the formula $F$ becomes the following set of two clauses:

$$x_1 + x_2 + x_3 \quad \text{and} \quad x_4 + \overline{x_1} + x_2.$$

Note, that Max 3SAT is a maximization problem.

Definition 1.2.9. Algorithm Alg for a maximization problem achieves an approximation factor $\alpha$ if for all inputs, we have:

$$\frac{\text{Alg}(G)}{\text{Opt}(G)} \geq \alpha.$$

In the following, we present a randomized algorithm – it is allowed to consult with a source of random numbers in making decisions. A key property we need about random variables, is the linearity of expectation property, which is easy to derive directly from the definition of expectation.

Definition 1.2.10 (*Linearity of expectations*). Given two random variables $X, Y$ (not necessarily independent, we have that $\mathbb{E}[X + Y] = \mathbb{E}[X] + \mathbb{E}[Y]$.

Theorem 1.2.11. One can achieve (in expectation) $(7/8)$-approximation to Max 3SAT in polynomial time. Namely, if the instance has $m$ clauses, then the generated assignment satisfies $(7/8)m$ clauses in expectation.
Proof: Let $x_1,\ldots,x_n$ be the $n$ variables used in the given instance. The algorithm works by randomly assigning values to $x_1,\ldots,x_n$, independently, and equal probability, to 0 or 1, for each one of the variables.

Let $Y_i$ be the indicator variables which is 1 if (and only if) the $i$th clause is satisfied by the generated random assignment and 0 otherwise, for $i = 1,\ldots,m$. Formally, we have

$$Y_i = \begin{cases} 1 & C_i \text{ is satisfied by the generated assignment,} \\ 0 & \text{otherwise.} \end{cases}$$

Now, the number of clauses satisfied by the given assignment is $Y = \sum_{i=1}^m Y_i$. We claim that $\mathbb{E}[Y] = (7/8)m$, where $m$ is the number of clauses in the input. Indeed, we have

$$\mathbb{E}[Y] = \mathbb{E}\left[ \sum_{i=1}^m Y_i \right] = \sum_{i=1}^m \mathbb{E}[Y_i]$$

by linearity of expectation. Now, what is the probability that $Y_i = 0$? This is the probability that all three literals appear in the clause $C_i$ are evaluated to $\text{FALSE}$. Since the three literals are instance of three distinct variable, these three events are independent, and as such the probability for this happening is

$$\mathbb{P}[Y_i = 0] = \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{8}.$$ 

(Another way to see this, is to observe that since $C_i$ has exactly three literals, there is only one possible assignment to the three variables appearing in it, such that the clause evaluates to $\text{FALSE}$. Now, there are eight (8) possible assignments to this clause, and thus the probability of picking a $\text{FALSE}$ assignment is 1/8.) Thus,

$$\mathbb{P}[Y_i = 1] = 1 - \mathbb{P}[Y_i = 0] = \frac{7}{8},$$

and

$$\mathbb{E}[Y_i] = \mathbb{P}[Y_i = 0] \cdot 0 + \mathbb{P}[Y_i = 1] \cdot 1 = \frac{7}{8}.$$ 

Namely, $\mathbb{E}[\text{# of clauses sat}] = \mathbb{E}[Y] = \sum_{i=1}^m \mathbb{E}[Y_i] = (7/8)m$. Since the optimal solution satisfies at most $m$ clauses, the claim follows. \hfill \blacksquare

Curiously, Theorem 1.2.11 is stronger than what one usually would be able to get for an approximation algorithm. Here, the approximation quality is independent of how well the optimal solution does (the optimal can satisfy at most $m$ clauses, as such we get a $(7/8)$-approximation. Curiouser and curiouser\footnote{“Curiouser and curiouser!” Cried Alice (she was so much surprised, that for the moment she quite forgot how to speak good English). – Alice in wonderland, Lewis Carol}. the algorithm does not even look on the input when generating the random assignment.

Håstad \cite{Hås01} proved that one can do no better; that is, for any constant $\varepsilon > 0$, one can not approximate $3\text{SAT}$ in polynomial time (unless $P = \text{NP}$) to within a factor of $7/8 + \varepsilon$. It is pretty amazing that a trivial algorithm like the above is essentially optimal.

Remark 1.2.12. For $k \geq 3$, the above implies $1 - 2^{-k}$-approximation algorithm, for $k$-\text{SAT}, as long as the instances are each of length at least $k$. 

\begin{thebibliography}{10}

\bibitem{Hås01} Håstad [Hås01] proved that one can do no better; that is, for any constant $\varepsilon > 0$, one can not approximate $3\text{SAT}$ in polynomial time (unless $P = \text{NP}$) to within a factor of $7/8 + \varepsilon$. It is pretty amazing that a trivial algorithm like the above is essentially optimal.

Remark 1.2.12. For $k \geq 3$, the above implies $1 - 2^{-k}$-approximation algorithm, for $k$-\text{SAT}, as long as the instances are each of length at least $k$. 

\end{thebibliography}
1.3. The Markov and Chebyshev’s inequalities

1.3.1. Markov’s inequality

We remind the reader that for a random variable $X$ assuming real values, its expectation is $\mathbb{E}[Y] = \sum_y y \cdot \mathbb{P}[Y = y]$. Similarly, for a function $f(\cdot)$, we have $\mathbb{E}[f(Y)] = \sum_y f(y) \cdot \mathbb{P}[Y = y]$.

**Theorem 1.3.1 (Markov’s Inequality).** Let $X$ be a random variable assuming only non-negative values. Then for all $C > 0$, we have $\mathbb{P}[X \geq C] \leq \frac{\mathbb{E}[X]}{C}$.

**Proof:** Indeed,

\[
\mathbb{E}[Y] = \sum_{y \geq 1} y \mathbb{P}[Y = y] + \sum_{y < t} y \mathbb{P}[Y = y] \geq \sum_{y \geq t} y \mathbb{P}[Y = y] \\
\geq \sum_{y \geq t} t \mathbb{P}[Y = y] = t \mathbb{P}[Y \geq t].
\]

Markov inequality is tight, as the following exercise testifies.

**Exercise 1.3.2.** For any (integer) $k > 1$, define a random positive variable $X_k$ such that $\mathbb{P}[X_k \geq k \mathbb{E}[X_k]] = 1/k$.

1.3.1.1. Another example for expectation

Let $X_i \in \{-1, +1\}$ with probability half for each value, for $i = 1, \ldots, n$ (all picked independently). Let $Y = \sum_i X_i$.

$$
\mathbb{E}[Y] = \mathbb{E} \left[ \sum_i X_i \right] = \sum_i \mathbb{E}[X_i] = n \cdot 0 = 0.
$$

A more interesting quantity is

$$
\mathbb{E}[Y^2] = \mathbb{E} \left[ \left( \sum_i X_i \right)^2 \right] = \mathbb{E} \left[ \sum_i X_i^2 + 2 \sum_{i < j} X_i X_j \right] = \sum_i \mathbb{E}[X_i^2] + 2 \sum_{i < j} \mathbb{E}[X_i X_j] = n + 2 \sum_{i < j} \mathbb{E}[X_i X_j] = n.
$$

**Lemma 1.3.3.** Let $X_i \in \{-1, +1\}$ with probability half for each value, for $i = 1, \ldots, n$ (all picked independently). We have that $\mathbb{P}[|\sum_i X_i| > t\sqrt{n}] < 1/t^2$.

**Proof:** Let $Y = \sum_i X_i$ and $Z = Y^2$. We have

$$
\mathbb{P}\left[ \left| \sum_i X_i \right| > t\sqrt{n} \right] = \mathbb{P}\left[ \left( \sum_i X_i \right)^2 > t^2 n \right] = \mathbb{P}[Y^2 > t^2 \mathbb{E}[Y^2]] = \mathbb{P}[Z > t^2 \mathbb{E}[Z]] \leq 1/t^2,
$$

by Markov’s inequality.
1.3.2. Chebychev’s inequality

Theorem 1.3.4 (Chebyshev’s inequality). Let \( X \) be a real random variable, with \( \mu_X = \mathbb{E}[X] \), and \( \sigma_X = \sqrt{\mathbb{V}[X]} \). Then, for any \( t > 0 \), we have \( \mathbb{P}[|X - \mu_X| \geq t\sigma_X] \leq 1/t^2 \).

Proof: Note that
\[
\mathbb{P}[|X - \mu_X| \geq t\sigma_X] = \mathbb{P}[(X - \mu_X)^2 \geq t^2\sigma_X^2].
\]
Set \( Y = (X - \mu_X)^2 \). Clearly, \( \mathbb{E}[Y] = \sigma_X^2 \). Now, apply Markov’s inequality to \( Y \). \( \square \)

1.4. Quick Sort

Let the input be a set \( T = \{t_1, \ldots, t_n\} \) of \( n \) items to be sorted. We remind the reader, that the QuickSort algorithm randomly pick a pivot element (uniformly), splits the input into two subarrays of all the elements smaller than the pivot, and all the elements larger than the pivot, and then it recurses on these two subarrays (the pivot is not included in these two subproblems). Here we will show that the expected running time of QuickSort is \( O(n \log n) \).

Definition 1.4.1. For an event \( \mathcal{E} \), let \( X \) be a random variable which is 1 if \( \mathcal{E} \) occurred and 0 otherwise. The random variable \( X \) is an indicator variable.

Observation 1.4.2. For an indicator variable \( X \) of an event \( \mathcal{E} \), we have
\[
\mathbb{E}[X] = 1 \cdot \mathbb{P}[X = 1] = \mathbb{P}[\mathcal{E}].
\]

Let \( S_1, \ldots, S_n \) be the elements in their sorted order (i.e., the output order). Let \( X_{ij} = 1 \) be the indicator variable which is one iff QuickSort compares \( S_i \) to \( S_j \), and let \( p_{ij} \) denote the probability that this happens. Clearly, the number of comparisons performed by the algorithm is \( C = \sum_{i<j} X_{ij} \). By linearity of expectations, we have
\[
\mathbb{E}[C] = \mathbb{E}\left[ \sum_{i<j} X_{ij} \right] = \sum_{i<j} \mathbb{E}[X_{ij}] = \sum_{i<j} p_{ij}.
\]

We want to bound \( p_{ij} \), the probability that the \( S_i \) is compared to \( S_j \). Consider the last recursive call involving both \( S_i \) and \( S_j \). Clearly, the pivot at this step must be one of \( S_i, \ldots, S_j \), all equally likely. Indeed, \( S_i \) and \( S_j \) were separated in the next recursive call.

Observe, that \( S_i \) and \( S_j \) get compared if and only if pivot is \( S_i \) or \( S_j \). Thus, the probability for that is \( 2/(j - i + 1) \). Indeed,
\[
p_{ij} = \mathbb{P}[S_i \text{ or } S_j \text{ picked } | \text{ picked pivot from } S_1, \ldots, S_j] = \frac{2}{j - i + 1}.
\]

Thus,
\[
\sum_{i=1}^{n} \sum_{j>i} p_{ij} = \sum_{i=1}^{n} \sum_{j>i} 2/(j - i + 1) = \sum_{i=1}^{n} \sum_{k=1}^{n-i+1} \frac{2}{k} \leq 2 \sum_{i=1}^{n} \sum_{k=1}^{n} \frac{1}{k} \leq 2nH_n \leq n + 2n \ln n,
\]
where \( H_n \) is the harmonic number\(^3\) \( H_n = \sum_{i=1}^{n} 1/i \), We thus proved the following result.

\(^3\)Using integration to bound summation, we have \( H_n \leq 1 + \int_{x=1}^{n} \frac{1}{x} \, dx \leq 1 + \ln n \). Similarly, \( H_n \geq \int_{x=1}^{n} \frac{1}{x} \, dx = \ln n \).
**Lemma 1.4.3.** QuickSort performs in expectation at most $n + 2n \ln n$ comparisons, when sorting $n$ elements.

Note, that this holds for all inputs. No assumption on the input is made. Similar bounds holds not only in expectation, but also with high probability.

This raises the question, of how does the algorithm pick a random element? We assume we have access to a random source that can get us number between 1 and $n$ uniformly.

Note, that the algorithm always works, but it might take quadratic time in the worst case.

**Remark 1.4.4 (Wait, wait, wait).** Let us do the key argument in the above more slowly, and more carefully. Imagine, that before running QuickSort we choose for every element a random priority, which is a real number in the range $[0, 1]$. Now, we reimplement QuickSort such that it always pick the element with the lowest random priority (in the given subproblem) to be the pivot. One can verify that this variant and the standard implementation have the same running time. Now, $a_i$ gets compares to $a_j$ if and only if all the elements $a_i+1, \ldots, a_{j-1}$ have random priority larger than both the random priority of $a_i$ and the random priority of $a_j$. But the probability that one of two elements would have the lowest random-priority out of $j - i + 1$ elements is $2 \times \frac{1}{j - i + 1}$, as claimed.

**Bibliography**