Social Networks and Social Choice **Lecture Date: September 7, 2010**

Lecture: Aggregation of General Biased Signals

Lecturer: Elchanan Mossel *Scribe: Miklos Racz*

So far we have talked about Condorcet's Jury Theorem [6] and its extensions in situations where the signals were binary. In this lecture we look at extensions where the signals are not binary.

1 Decision between k alternatives

First, let us consider the case when n people make a decision between k alternatives, denoted $1, \ldots, k$, where one of the k alternatives is correct. We can make various assumptions on the type of signals individuals receive. The most general assumption is the following: there is a signal space X (e.g. $X = \mathbb{R}$), and distributions $P_1, \ldots, P_k$ on X, and if the correct alternative is i, then each individual receives independently a signal distributed according to P_i. A nice non-general assumption that we analyze in more detail is this: each individual receives independently the correct alternative with probability $p > 1/k$, and each other alternative with probability $(1-p)/(k-1)$.

1.1 Plurality functions

Consider this latter setup; what should the aggregation function be? We now introduce the concept of a plurality function and show that it is a good aggregation function; in fact, we will show below that in a sense it is the best among fair aggregation functions. Let $n_a(x)$ denote the number of a's in the vector x. A function $f : [k]^n \to [k]$ is a plurality function if $f(x) = a$ if $n_a(x) > n_b(x)$ for all $b \neq a$. Note that ties in this case cannot be avoided (as opposed to the two alternatives case with n being odd), and so there are many plurality functions. A function $f : [k]^n \to [k]$ is fair if for all $\sigma \in S_k$, we have $f(\sigma x) := f(\sigma x_1, \ldots, \sigma x_k) = \sigma f(x)$. Note that fairness corresponds to treating all alternatives equally—their names do not matter. For example, if f is a plurality function, $n = 5, k = 3$, and $f(2,2,3,3,1) = 2$, then fairness of f means that $f(3,3,2,2,1) = 3$, $f(1,1,3,3,2) = 1$, etc.

[Note that now there are two ways permutations act in our setup. First, we can permute the alternatives: $x \to \sigma x$ for $\sigma \in S_k$. Symmetry with respect to the alternatives corresponds to the notion of fairness. Second, we can permute voters: $x \to x_\sigma$ for $\sigma \in G$ where G is a group acting transitively on $[n]$. Symmetry with respect to the voters corresponds to the notion of transitivity.]

The following exercises are all homework:

- Prove that for all n and k every plurality function is monotone. (Monotonicity in the case of more than two alternatives is defined in Section 3.)
- Prove that for all n and k there exists a fair plurality function.
- Prove that for all n and k there exists a transitive plurality function.

The following theorem (an analogue of the similar theorem for the two alternatives case) shows that a plurality function is a good aggregation function.

Theorem 1.1. *Consider the previous setup, where each individual receives independently the correct alternative with probability $p > 1/k$, and each other alternative with probability $(1-p)/(k-1)$. Assume further a uniform prior. Write $c(n) = \mathbb{P}(\text{Plurality is correct})$. Then*

$$\lim_{n\to\infty} c(n) = 1.$$

The same is true even for $p(n)$ for which $p(n) - 1/k >> n^{-1/2}$. If $p(n) - 1/k << n^{-1/2}$ then $c(n) \to 1/k$. Furthermore, if we write $a(n) = p(n) - 1/k$, then for all n we have

$$c(n) \geq 1 - 2ke^{-a^2(n)n}.$$

Proof. Generalization of the binary case to k alternatives. □

In fact:

Theorem 1.2. *Plurality maximizes the probability of being correct among fair functions.*

Proof. Same as proof for majority. □

1.2 Estimation point of view

In a way this is a classical estimation problem: there is a random variable S with a uniform prior, and our goal is to estimate S given the signals $X_1, \ldots, X_n$. We choose the s which maximizes $\mathbb{P}(S = s|X_1, \ldots, X_n)$, but since the uniform is prior, this is the same as maximizing $\mathbb{P}(X_1, \ldots, X_n|S = s)$ (by Bayes' rule). The estimation point of view is valid also for the general signals picture: the optimal choice function chooses the s maximizing $\mathbb{P}(S = s|X_1, \ldots, X_n)$. One can apply general results from statistics to obtain similar results to the ones above, but one needs to think carefully about how to measure bias to obtain guarantees.

2 Two examples of more general signals

2.1 Possibility of staying at home

Now let us consider the possibility of staying at home. Suppose there are two alternatives, $+$ and $-$, and one votes for the correct alternative with probability p, one votes for the wrong alternative with probability $q < p$, and one does not vote with probability $1 - p - q \geq 0$. Assuming a uniform prior, what is the optimal aggregation function? It is again the majority, which can be shown by a likelihood calculation. The next question is how large should $p - q$ be to aggregate well? As before, $p - q >> n^{-1/2}$ suffices, but in this case this is not always necessary. Consider the degenerate case of $q = 0$: then all the majority needs in order to be correct is one person who goes and votes (since everyone who votes, votes correctly). In this case

$$\mathbb{P}(\text{nobody votes}) = (1-p)^n$$

and so for instance $p = \log(n)/n$ suffices for the majority to aggregate.

2.2 Decision between k alternatives based on rankings

Let us now consider the following setup. There are again k possible alternatives, but now each voter receives a ranking. In this ranking the correct alternative is at location i with probability p_i, and all other alternatives are placed uniformly at random. So a particular ranking which has the correct alternative at location i has probability $p_i/\left((k-1)!\right)$. We assume that $p_1 > \cdots > p_k$ (and of course $\sum_i p_i = 1$). Assuming a uniform prior, what is the optimal aggregation function? To answer this, let us calculate the log-likelihood of a sample.

$$\begin{aligned} S_i := \log \mathbb{P}\left(\text{rankings}|\text{intended winner is } i\right) = &-n\log\left((k-1)!\right) + \#\left\{\text{rankings where } i \text{ is top}\right\}\log p_1 \\ &+ \#\left\{\text{rankings where } i \text{ is } 2^{\text{nd}} \text{ place}\right\}\log p_2 \\ &+ \dots \\ &+ \#\left\{\text{rankings where } i \text{ is } k^{\text{th}} \text{ place}\right\}\log p_k. \end{aligned}$$

As discussed in Section 1.2, the optimal function chooses i which maximizes S_i. Another question is: what is the difference needed between the p_i's for aggregation?

Note: the voting method defined above is a generalization of the Borda count, a voting method developed by Jean-Charles de Borda (May 4, 1733 - February 19, 1799), who was a French mathematician, physicist, political scientist, and sailor [3]. In the Borda count, for every vote the alternative voted first place gets k points, the alternative voted second place gets $k-1$ points, and so on, until finally the alternative voted last gets 1 point. The points of each alternative are summed up over all votes, and whoever has the most points wins the election.

3 Beyond the plurality function

What can we say about other aggregation functions, e.g. the U.S. electoral college? We again assume simple signals: each individual receives independently the correct alternative with probability $p > 1/k$, and each other alternative with probability $(1-p)/(k-1)$. Again we wish to consider functions that are (i) fair—names of alternatives do not matter, and (ii) monotone—a stronger vote in one direction should not hurt this direction. We defined fairness in Section 1.1, let us now define monotonicity. We first define two types of monotonicity among vectors, one stronger than the other. For two vectors $x, y \in [k]^n$ and for $a \in [k]$ write $x <_a y$ to indicate that $y_i = a$ holds whenever $x_i \neq y_i$ ("y is leaning more towards a than x"). For two vectors $x, y \in [k]^n$ and for $a \in [k]$ write $x \leq_a y$ if $x_i = a$ implies $y_i = a$. Certainly $x <_a y$ implies $x \leq_a y$. We say that a function $f : [k]^n \to [k]$ is monotone if for all $a \in [k]$ and all $x, y \in [k]^n$, $x <_a y$ implies $f(x) \leq_a f(y)$. So if f is monotone, a wins for vote x and $x <_a y$, then a also wins for vote y. The following result gives sufficient conditions for aggregation.

Theorem 3.1 (Kalai-Mossel)**.** *Suppose $f : [k]^n \to [k]$ is a monotone transitive aggregation function where for $p = 1/k$ it holds that $\mathbb{P}(f = a) \geq 1/2k$ for all a. Then there exists $c = c(k)$ such that for every $\epsilon < 1/3$ it holds that for*

$$p > \frac{1}{k} + c\frac{\left(\log(1-\epsilon) - \log(1/2k)\right)\log\log n}{\log n}$$

we have

$$\mathbb{P}\left(f \text{ is correct}\right) \geq 1 - \epsilon.$$

We do not prove this theorem. Note that this shows aggregation in the case of the electoral college with all states of equal sizes, and for plurality functions, though in these cases even a smaller bias of $Cn^{-1/2}$ suffices.

4 Additional Structure

So far we have assumed that the different alternatives and signals have no additional structure. We now consider two examples of such structures. The first example deals with signals that are real numbers, while the second example deals with signals that are rankings.

4.1 Signals that are real numbers

Let us look at the following setup. Assume that the true state of the world is either $s = +1$ or $s = -1$. Each voter receives independently a real-valued signal distributed according to $N(as, 1)$ where $a > 0$ is some constant. How should people vote? One option is for voters who got a positive signal to vote $+1$ and voters who got a negative signal to vote -1, and then take a majority vote. This is a pretty good voting method; for example $a >> n^{-1/2}$ suffices to get the correct answer with probability tending to 1. But it is not optimal. The best Bayesian decision rule (assuming a (1/2, 1/2) uniform prior) is the following: each voter declares their signal X_i and the outcome of the vote is the sign of $\sum_i X_i$. Note however, that this voting rule lets one cheater determine the outcome of the election: the cheater just says a number with very large absolute value and with the desired sign. The majority vote of the signs of the signals is more robust.

In general, we have the following theorem.

Theorem 4.1 (Keller-Mossel-Sen [8]). *If $f : \mathbb{R}^n \to \{-1, +1\}$ is a monotone transitive function with $\mathbb{E}_{a=0}(f) = 0$, then $\mathbb{E}_a(f) \to 1$ if $a >> (\log n)^{-1/2}$.*

So any democratic function would work. However, non-democratic functions (e.g. dictator, functions of a few voters) will not aggregate even for a constant a.

4.2 Signals that are rankings

Now let us discuss n voters who rank k alternatives, and the outcome should be a ranking of the k alternatives. A good question is: what is a good model here?

4.2.1 Plurality vote

Should we use a plurality vote? This may not be a good idea for two reasons.

First of all, there are computational considerations. If we have k alternatives then there are $k!$ possible rankings, which grows very fast in k. Consider a distribution on the rankings where the true permutation is twice as likely as any other permutation; if we apply plurality rule we may need on the order of $k!$ voters to get a good answer. If k is large (say even $k = 100$), then this is too big.

Second, it may happen that a very large percentage of voters ranks some alternative at the top—in which case it is a "no-brainer" to rank this alternative at the top—but the plurality function does not do this. Consider the following example. There are $(k-1)! + 2$ voters, the first $(k-1)!$ all give a different ranking, but all of them rank alternative 1 at the top, and the last two voters give the same ranking, in which alternative 1 is not at the top. A plurality function would have this latter ranking as its outcome, not ranking alternative 1 at the top.

4.2.2 Consensus ranking

Instead of a plurality vote, another natural output to consider is the consensus ranking. Given a set of rankings $\{\pi_1, \ldots, \pi_n\}$, the consensus ranking is the following "average":

$$\pi_0 = \operatorname{argmin} \sum_{i=1}^{n} d(\pi_i, \pi_0),$$

where d is a distance on S_n, the set of permutations of n objects. The most natural distance to consider is the Kendall distance d_K, given by

$$d_K(\pi, \sigma) = \#\{(i,j) : \pi(i) < \pi(j) \text{ and } \sigma(i) > \sigma(j)\}.$$

A historical note: Maurice Kendall (6 September 1907 - 29 March 1983) was a British statistician; Kendall's τ rank statistic uses the Kendall distance to test if two variables are statistically independent.

4.2.3 Mallows' model

Mallows' model, introduced by Colin Mallows in [10], is a classical model for noisy permutations. This is an exponential family model in β—given that the "true" ranking is π_0, the probability of observing a permutation π is exponentially small in β times the Kendall distance between π and π_0:

$$\mathbb{P}(\pi|\pi_0) = \frac{1}{Z(\beta)} e^{-\beta d_K(\pi, \pi_0)},$$

where $\beta > 0$ and $Z(\beta)$ is a normalizing constant. Assuming a uniform prior on the true ranking, the maximum likelihood estimator of the true ranking given rankings $\pi_1, \ldots, \pi_n$ is the consensus ranking defined above.

Assume $\beta > 0$ is a fixed constant. How many voters are needed in order to be able to recover the true ranking with good probability? Algorithmically: how can one find the consensus ranking of given rankings $\pi_1, \ldots, \pi_n$? These are natural questions to ask regarding the model, and we now provide some pointers to the literature dealing with these questions. Bartholdi, Tovey and Trick showed in [2] that the optimization problem of finding the consensus ranking of given rankings is NP-hard. Nevertheless, several papers deal with approximations to the solution. Cohen, Schapire and Singer provide simple and efficient greedy algorithms that are guaranteed to find a good approximation (in fact, a 2-approximation) to the true ranking [5]. Ailon, Charikar and Newman provide a randomized algorithm that gives an expected 11/7-approximation [1]. Kenyon-Mathieu and Schudy provide a polynomial time approximation scheme for the problem, i.e. for every $\epsilon > 0$ they provide a $(1+\epsilon)$-approximation which runs in polynomial time in the number of alternatives that are ranked [9]. Meila, Phadnis, Patterson and Bilmes provide a branch and bound algorithm that gives back the consensus ranking exactly; the algorithm has factorial running time in the worst case, but it can do better in some cases [11].

Let us now discuss the recent paper by Braverman and Mossel [4] in more detail, which gives a randomized polynomial time algorithm for computing the consensus ranking exactly with high probability. Let us change the notation to follow that which is used in [4]: let n denote the number of possible alternatives, and let r denote the number of sample rankings (i.e. the number of voters). The main result is the following.

Theorem 4.2 (Braverman-Mossel [4])**.** *Given r independent samples from the Mallows model, $\pi_1, \ldots, \pi_r$, there exists a randomized polynomial time algorithm which finds the maximum likelihood solution (i.e. the consensus ranking) τ exactly with high probability (say at least $1 - n^{-100}$). The running time of the algorithm is $n^{1+O\left((\beta r)^{-1}\right)}$; in particular, the running time tends to almost linear as r grows. The query complexity of the algorithm is $O(n \log n)$ (with the constant depending on β and r).*

The proof consists of two ingredients. The first ingredient is concerned with the statistical properties of the generated permutations π_i in terms of the true order π_0. Then with high probability it holds that

$$\sum_i |\pi_0(i) - \tau(i)| = O(n/(\beta r)), \quad \text{and} \quad \max_i |\pi_0(i) - \tau(i)| = O(\log(n)/(\beta r)).$$

This is important in order to show that if we have a solution π^* which is close to the true ranking π_0, then it is also close to the consensus ranking τ. We can arrive at such a π^* ordering by "averaging": taking the average of the locations of element x under the π_i's and ranking the elements according to this locates each element within a distance of $O\left(\frac{1}{\beta r}\log n\right)$ from its location in the true order π_0 with high probability.

The second ingredient is a dynamic programming algorithm which finds τ given a starting point where each element is at most k away, with running time $O\left(nk^2 2^{6k}\right)$ (see [4, Lemma 16]). In our case $k = O(\log n)$, giving a polynomial time algorithm.

The algorithm goes as follows. First notice the following. If $i < j$ are indices, $I = [i, j]$, then the set of elements $S_I = \{\tau(i), \ldots, \tau(j)\}$ can be found amongst the elements found at indices in the interval $[i-k, j+k]$, and the elements found at indices in the interval $[i+k, j-k]$ are surely amongst the elements in S_I. So selecting the set $S_I = \{\tau(i), \ldots, \tau(j)\}$ involves selecting $2k$ elements from the elements found at indices in $[i-k, \ldots, i+k-1] \cup [j-k+1, j+k]$, thus the number of possible S_I's is bounded by $\binom{4k}{2k} \leq 2^{4k}$.

Without loss of generality suppose that n is a power of 2, and divide $[n]$ up into intervals by halving: first into 2 intervals of length $n/2$, then 4 intervals of length $n/4$, and so on, finally into $n/2$ intervals of length 2. In total we have $n-1$ intervals of length $2, 4, 8, \ldots$. Now starting from intervals of smaller length, store optimal permutations of all variations of possible elements on a given interval. For each interval there are at most 2^{4k} variations of possible elements, and we have $n-1$ intervals altogether, so the storage needed is polynomial in n if $k = O(\log n)$. The question now is that if we have two adjacent intervals of length l that are pre-sorted, then how do we sort the union of these two intervals. The observation is that the "middle" of the two respective intervals will remain unchanged, and we only need to go through the possibilities of swapping elements at the "edge" of the two intervals where they meet (which is the "middle" of the big interval). (See lecture slides for an illustration.) In the end, the runtime can be calculated to be

$$\sum_{i=1}^{\log n} \#\text{intervals of length } 2^i \cdot \#\text{checks} \cdot \text{cost/check} = \sum_{i=1}^{\log n} O\left(\frac{n \cdot 2^{4k}}{2^i} \cdot 2^{2k} \cdot k^2\right) = O\left(nk^2 2^{6k}\right).$$

4.2.4 Some notes on related problems

Disregarding its social context, the problem above is an example of sorting from noisy information. Here are two more examples of a similar form.

Consider a sorting problem where for each query comparing two elements x and y, the correct answer is returned with probability $1/2 + \epsilon$, independently for each query. Each pair can be queried as many times as we want. How many queries are needed to find the correct order with probability 0.9999? If there is no randomness, then $O(n \log n)$ queries are needed, using any of the standard comparison sort algorithms. If there is randomness, then repeating each query $O\left(\epsilon^{-2} \log n\right)$ times we can determine the order between each pair with high probability. However, using more sophisticated methods the true order can be found in query complexity $O(n \log n)$ with high probability, see [7].

Consider a sorting problem where for each query comparing two elements x and y, the correct answer is returned with probability $1/2+\epsilon$, independently for each query. However, each pair can be queried only once now. Braverman and Mossel show in [4] a randomized polynomial time algorithm that finds the maximum likelihood solution with high probability.

References

[1] N. Ailon, M. Charikar, and A. Newman. Aggregating Inconsistent Information: Ranking and Clustering. *Journal of the ACM (JACM)*, 55(5):1–27, 2008.

[2] J. Bartholdi, III, C. A. Tovey, and M. A. Trick. Voting schemes for which it can be difficult to tell who won the election. *Social Choice and Welfare*, 6:157–165, 1989. 10.1007/BF00303169.

[3] J.C. Borda. Mémoire sur les élections au scrutin. *Histoire de l'Académie Royale des Sciences*, 2:85, 1781.

[4] M. Braverman and E. Mossel. Sorting from noisy information. Arxiv 0910.1191, 2010.

[5] W.W. Cohen, R.E. Schapire, and Y. Singer. Learning to Order Things. *Journal of Artificial Intelligence Research*, 10:243–270, 1999.

[6] M. Condorcet. Essai sur l'application de l'analyse à la probabilité des décisions rendues à la pluralité des voix. 1785.

[7] U. Feige, P. Raghavan, D. Peleg, and E. Upfal. Computing with noisy information. *SIAM Journal on Computing*, 23(5):1001–1018, 1994.

[8] N. Keller, E. Mossel, and A. Sen. Geometric influences. Arxiv 0911.1601, 2010.

[9] C. Kenyon-Mathieu and W. Schudy. How to rank with few errors. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of Computing*, pages 95–103. ACM, 2007.

[10] C.L. Mallows. Non-null ranking models. I. *Biometrika*, 44(1-2):114–130, 1957.

[11] M. Meila, K. Phadnis, A. Patterson, and J. Bilmes. Consensus ranking under the exponential model. In *Proceedings of the Twenty-Third Annual Conference on Uncertainty in Artificial Intelligence*, pages 285–294, Corvallis, Oregon, 2007. AUAI Press.