SMT SCORING SYSTEM

This document describes the scoring system for the Stanford Math Tournament. We begin by giving an overview of the changes to scoring and a non-technical description of the scoring rules. Then, we give a precise technical description of the scoring system.

1. Summary of scoring changes

We will normalize the scores on each test by a new method based on simultaneously estimating the relative performance of each contestant and the difficulty of each problem from the actual results of the tournament. This changes the way overall team scores are computed to make them fairer, by making scores on different tests more comparable. However, it does not affect the rankings on each separate test.

For test-taking strategy, the important point is that your score on each test still depends only on the number of problems you solve correctly. So the goal remains simple: solve as many problems as you can.

2. Scoring rules

On the individual tests and the Team Test, each problem is worth one point towards the *raw score*. (See the following sections for why these are the raw scores.) On the Power Round, problems are weighted, and the point value of each problem towards the score will be indicated on the test.

While scoring of each test is straightforward, determining how much each test should contribute to a team's overall score is a more complicated problem. For each test except the Power Round, the scores are normalized by the score normalization procedure described in the following sections. This converts each *raw score* to a *normalized score*.

Each *normalized score* is then multiplied by a weight depending on the test, to produce a *weighted score*. The weights determine how much each test is worth compared to the others. The weight for the individual tests (both general and subject tests) is 50, and the weight for the team test and power round is 400. The weighted scores on the Power round are simply the raw scores, divided by the maximum score possible score, and multiplied by the weight of 400.

A team's overall score is the sum of the weighted scores on the team test, the power round, and all of the team members' individual tests. So, individual tests count for about 50% of a team's overall score, and the team and power tests each count for about 25%. Note that the easier general test is worth approximately 50% as much as two subject tests.

3. Overview of score normalization

3.1. **Motivation.** Our scoring needs to produce two end results: a ranking of the competitors (individuals or teams) on each test, and an overall score for each team, which is a sum of contributions from the tests the team took. We therefore wish to assign a score to each competitor on each test which represents their relative performance on that test. These scores are the contributions to the overall team scores, and also determine the rankings on each test.

Because the overall team scores are composed of contributions from several different tests, the scores need to be normalized to be comparable across tests. Using the number of problems solved correctly or a sum of points awarded for each problem as the score is clearly not a good choice. The most obvious example is that individuals can take different subject tests or the general test, which vary in difficulty. Without any normalization, there would be a bias in favor of easier subject tests.

3.2. **Score normalization method.** The basic idea of our method for normalizing scores on each test is that we simultaneously estimate the relative performance of each contestant and the difficulty of each problem based on the actual results on the test. We then estimate the number of problems each contestant would solve on an idealized model test from the estimate of their relative performance. This is the contestant's normalized score. Note that scores on each test are normalized independently of the other tests.

A little more technically, we model the probability of a correct/incorrect response by each competitor on each problem as a function of parameters for the competitor's ability and the problem's difficulty. We estimate these parameters by finding the values that best fit the results of the tournament.

We then consider a model test where the problem difficulties are distributed according to a fixed probability distribution. Using the estimates of a competitor's ability, we compute the probability of the competitor solving a problem on this test.

This expected score is the normalized score we use. The overall team scores are sums of these scores, and this score normalization method yields scores that are more comparable between tests and more fair as contributions to the overall team scores.

One property of this model is that a competitor's normalized score depends only on the number of problems they solved correctly, not on which problems they solved. (This follows from the technical details, as described in the following sections.) Of course, the normalized score is higher for competitors who solve more problems. So, despite the complexity of the scoring system, the goal of each competitor is still simply to solve as many problems as they can. Also, the rankings of competitors on each test simply depend on the number of problems they solved correctly.

3.3. **Advantages.** In previous years, we normalized scores by dividing the scores on each test by the top ten average. This does help compared to just using raw, unnormalized scores, but these scores are still not very fair. For instance, this does not normalize for differences in variance. Consider a simple example: if two tests have the same top ten average but different variances, scores on the high-variance test become more important for overall team scores. Another flaw with this normalization scheme is that it focuses on normalizing scores for the top ten competitors, but does little to ensure that scores for competitors in other score brackets are accurately normalized.

The new scoring method helps resolve these flaws and other issues. It takes into account aspects of the score distribution other than just the top ten average, producing scores that are accurately normalized in all score brackets.

The only real disadvantage of the new scoring system is the greatly increased complexity involved in calculating overall team scores. However, this complexity has little direct impact on contestants, because the goal when taking a test remains essentially unchanged. We believe that the improvements in the fairness of scoring outweigh the loss in simplicity.

4. Score normalization technical details

We now discuss the technical details of the score normalization method.

4.1. **Model.** To normalize scores on a test, we model the probability of the response of each competitor on each problem. Our model has the following parameters: the ability score α_c of each competitor c and the difficulty δ_p of each problem p. We model the probability that a competitor with a given ability score α_c correctly solves a problem with a given difficulty δ_p as $g(\alpha_c - \delta_p)$, where $g(x) = \frac{e^x}{1+e^x}$ is a logistic function. (This is a Rasch model, or a 1-parameter item response theory model.)

We will compute the MAP estimate, and assign each competitor's score based on the MAP estimate of their ability score $\widehat{\alpha}_c$.

For each competitor c, α_c is a priori distributed as an (improper) uniform distribution over $\mathbb{R}$ (or, equivalently, there is no prior).

The problem difficulties $\vec{\delta}$ are a priori distributed as

$$P(\vec{\delta}) = \prod_p \exp\left(-\frac{(\delta_p - \overline{\delta})^2}{2\sigma_\delta^2}\right)$$

over $\mathbb{R}$ (with appropriate normalization), where $\overline{\delta}$ is the mean of the δ_p, and $\sigma_\delta = 5$. This is a Gaussian around $\overline{\delta}$ for each problem.

We assume that the observed results on each problem by each competitor are conditionally independent of each other given the parameters. Thus, the likelihood function is

$$P(\mathcal{D} \mid \vec{\alpha}, \vec{\delta}) = \prod_{c \text{ correct on } p} g(\alpha_c - \delta_p) \prod_{c \text{ incorrect on } p} (1 - g(\alpha_c - \delta_p)).$$

4.2. **Parameter estimation.** We wish to maximize the posterior probability

$$P(\vec{\alpha}, \vec{\delta} \mid \mathcal{D}) = \frac{P(\mathcal{D} \mid \vec{\alpha}, \vec{\delta}) P(\vec{\alpha}, \vec{\delta})}{P(\mathcal{D})}.$$

Dropping the constant $P(\mathcal{D})$ and normalization constants, this is equivalent to maximizing

$$\prod_{c \text{ correct on } p} g(\alpha_c - \delta_p) \prod_{c \text{ incorrect on } p} (1 - g(\alpha_c - \delta_p)) \prod_p \exp\left(-\frac{(\delta_p - \overline{\delta})^2}{2\sigma_\delta^2}\right).$$

This is equivalent to maximizing its logarithm

$$\sum_{c \text{ correct on } p} \log g(\alpha_c - \delta_p) + \sum_{c \text{ incorrect on } p} \log(1 - g(\alpha_c - \delta_p)) - \frac{1}{2\sigma_\delta^2} \sum_p \left(\delta_p - \overline{\delta}\right)^2.$$

For convenience, let

$$l(\vec{\alpha}, \vec{\delta}) = \sum_{c \text{ correct on } p} \log g(\alpha_c - \delta_p) + \sum_{c \text{ incorrect on } p} \log(1 - g(\alpha_c - \delta_p))$$

be the log-likelihood.

Note that the posterior probability is invariant under an arbitrary translation of all the parameters. Our model does not choose any origin for the scale on which the parameters α_c and δ_p are located. We will discuss this later.

We can given an equivalent definition of this optimization problem. Suppose we replace the prior on $\vec{\delta}$ with

$$\prod_p \exp\left(-\frac{(\delta_p - \mu_\delta)^2}{2\sigma_\delta^2}\right)$$

where μ_δ is a constant. This replaces the term $\sum_p(\delta_p - \overline{\delta})^2$ with $\sum_p(\delta_p - \mu_\delta)^2$.

We claim that a solution maximizing the new log posterior $l - \frac{1}{2\sigma_\delta^2}\sum_p(\delta_p - \mu_\delta)^2$ has $\overline{\delta} = \mu_\delta$. This is because given any solution, we can add a constant a to all the parameters without changing the likelihood, so we wish to set a to minimize $\sum_p(\delta_p + a - \mu_\delta)^2$. Take

$$\frac{d}{da}\sum_p(\delta_p + a - \mu_\delta)^2 = \sum_p \delta_p + a - \mu_\delta = 0.$$

So the optimum is at $a = \mu_\delta - \overline{\delta}$. So, a solution with $\overline{\delta} \neq \mu_\delta$ cannot be a maximum, because we can increase the log posterior by shifting the parameters such that $\overline{\delta} = \mu_\delta$.

Because of this, it is easy to see that maximizing $l - \frac{1}{2\sigma_\delta^2}\sum_p(\delta_p - \mu_\delta)^2$ is equivalent to maximizing $l - \frac{1}{2\sigma_\delta^2}\sum_p(\delta_p - \overline{\delta})^2$, up to a translation of all the parameters.

So, we now wish to maximize

$$F(\vec{\alpha}, \vec{\delta}) = \sum_{c \text{ correct on } p} \log g(\alpha_c - \delta_p) + \sum_{c \text{ incorrect on } p} \log\left(1 - g(\alpha_c - \delta_p)\right) - \frac{1}{2\sigma_\delta^2}\sum_p\left(\delta_p - \mu_\delta\right)^2.$$

We take the partial derivatives and set them equal to 0. Note that $\frac{d}{dx}g(x) = g(x)\left(1 - g(x)\right)$. For any competitor c, taking the partial derivative with respect to α_c yields

$$\begin{aligned} 0 = \frac{\partial F}{\partial \alpha_c} &= \sum_{c \text{ correct on } p} \left(1 - g(\alpha_c - \delta_p)\right) - \sum_{c \text{ incorrect on } p} g(\alpha_c - \delta_p) \\ &= r_c - \sum_p g(\alpha_c - \delta_p) \end{aligned}$$

where r_c is the number of problems that c answered correctly.

For any problem p, taking the partial derivative with respect to δ_p yields

$$\begin{aligned} 0 = \frac{\partial F}{\partial \delta_p} &= -\sum_{c \text{ correct on } p} \left(1 - g(\alpha_c - \delta_p)\right) + \sum_{c \text{ incorrect on } p} g(\alpha_c - \delta_p) - \frac{\delta_p - \mu_\delta}{\sigma_\delta^2} \\ &= -S_p + \sum_c g(\alpha_c - \delta_p) - \frac{\delta_p - \mu_\delta}{\sigma_\delta^2} \end{aligned}$$

where S_p is the number of contestants that answered p correctly.

Note from the equation for α_c that the score α_c depends only on the number of problems c solved correctly, given fixed difficulty parameters $\vec{\delta}$. So all contestants that solve the same number of problems have the same score, no matter which problems they solved. (In other words, r_c is a sufficient statistic.)

4.3. **An estimation algorithm.** Next, we describe an iterative algorithm for finding the maximum. (This is a hard EM algorithm, and can also be viewed as a coordinate ascent algorithm.) Note that $\frac{\partial F}{\partial \alpha_c}$ depends only on the parameters α_c and $\vec{\delta}$, and is monotonic in α_c. So given $\vec{\delta}$, we can solve the equation for α_c by binary search. Similarly, $\frac{\partial F}{\partial \delta_p}$ depends only on δ_p and $\vec{\alpha}$, and is monotonic in δ_p, so we can solve its equation by binary search given $\vec{\alpha}$. We can start by choosing an arbitrary $\vec{\alpha}^{(0)}$. Given a parameter vector $\vec{\alpha}^{(n)}$, we can set $\vec{\delta}^{(n)}$ as the solution to the equations for δ_p given $\vec{\alpha}^{(n)}$, and then set $\vec{\alpha}^{(n+1)}$ as the solution to the equations for α_c given $\vec{\delta}^{(n)}$. Note that each step is guaranteed to increase F.

4.4. **Concavity.** We now show that F is concave. Because of this, we can see that our algorithm is guaranteed to converge to the unique global maximum.

Proposition 4.1. *F is concave.*

Proof. We will first show that $\log g(\alpha - \delta)$ and $\log(1 - g(\alpha - \delta))$ are concave functions.

The Hessian of $\log g(\alpha - \delta)$ is

$$H = \begin{pmatrix} h(\alpha - \delta) & -h(\alpha - \delta) \\ -h(\alpha - \delta) & h(\alpha - \delta) \end{pmatrix}$$

where

$$h(\alpha - \delta) = -\frac{e^{\alpha - \delta}}{(1 + e^{\alpha - \delta})^2}.$$

Note that $h(x) \leq 0$ for all x.

For any z,

$$z^T H z = h(\alpha - \delta) z^T \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} z = h(\alpha - \delta)\left(z_1^2 - 2z_1 z_2 + z_2^2\right) = h(\alpha - \delta)(z_1 - z_2)^2 \leq 0.$$

Therefore, H is negative semidefinite, and $\log g(\alpha - \delta)$ is concave.

The Hessian of $\log(1 - g(\alpha - \delta))$ is the same, so this function is also concave.

The terms of F associated with the priors are single-variable quadratics which are obviously concave. Since F is a sum of concave functions, F is concave. □

4.5. **Computing scores.** We have described how we compute estimates of α_c for each contestant (up to a translation of all the parameters, which we will discuss in the next section). Next, we consider a model test where the problem difficulties are distributed as a Gaussian with mean 0 and standard deviation $\sigma = 2.5$:

$$P(\delta_p) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{\delta_p^2}{2\sigma^2}\right).$$

We will compute the expected number of problems a contestant with ability α_c solves on this test. This is just the probability of solving a problem times the number of problems, so from now on we will consider the model test to have one problem. The probability of solving a problem distributed as $P(\delta)$ is

$$\int_{-\infty}^{\infty} g(\alpha_c - \delta) P(\delta)\, d\delta.$$

This value is the *normalized score* of competitor c.

Note that the normalized score is in $[0, 1]$, with a normalized score of 0 when the competitor solves no problems, and a normalized score of 1 when the competitor solves all the problems.

4.6. **Setting the scale origin.** We now return to the issue of choosing a particular translation of the parameter estimates, because translating all the parameter estimates by a constant does not change the posterior probability in the scoring model. We wish to set a scale origin for the parameters of each test in such a way that scores are comparable across tests. There is no canonical way to do this, because we do not have a way to directly compare scores between tests.

One approach is to set the scale origin by assuming that the problem difficulties δ_p are distributed similarly on the tests. Consider taking Gaussian priors on δ_p with mean $\mu_\delta = 0$. These are the priors we used for the purposes of parameter estimation, which we found gave us an equivalent MAP estimate, up to translation by a constant. This is equivalent to setting $\overline{\delta} = 0$. Note that this centers the distribution the same location as the model test's difficulty distribution.

Another approach is to base our choice on an assumption that the contestant scores are distributed similarly on the tests. Consider taking an average of the contestant scores, and setting that equal to a fixed value. In particular, we will translate the parameters such that the mean *normalized score* of the middle half of contestants (between the first and third quartiles) is equal to a constant μ_s.

The approach based on the mean problem difficulty works reasonably well in general, yielding normalized scores that reflect the distribution of problem difficulties on the test, unlike the raw scores. The general and team tests have very different contestant populations, and thus score distributions, compared to each other and to the subject tests. Because of this, it does not make much sense to choose the scale origin for the general and team tests based on the contestant scores.

In contrast, for the subject tests, the assumption that the contestant scores are similarly distributed is fairly accurate. Using the contestant-based approach yields scores that are more accurate across the subject than just setting $\overline{\delta} = 0$, because the distribution of problem difficulties does differ significantly between subject tests. We therefore set the mean of the middle half of contestants to $\mu_s = 0.2$. This value is chosen empirically based on past results.

This method works to give scores on the subject tests that are fair and comparable between different subjects, even when they have different problem difficulties. The resulting parameter estimates tend to have $\overline{\delta}$ close to 0, so they are still comparable to the general and team test scores.