One sensible objection to the current team balancing algorithm is that it produces teams that are very different from each other, even if the mean skill is similar. That is, they might have very similar mean skill, but the standard deviation could be very different.

The ideal balancing algorithm would produce teams that are “indistinguishable.” Being indistinguishable actually has a precise definition in information theory, Mutual Information. If the mutual information between two random variables, x and y, is low, it means that learning x doesn’t tell you anything about the distribution of y. To translate that into our context, it means that learning which team a player is on shouldn’t tell you anything about the player’s skill, and learning the player’s skill shouldn’t tell you anything about which team they are on.

Starting with the definition of mutual information where s is the continuous random variable of the skill, and i is the discrete random variable of the team, the mutual information is

$$I(i;s) = \sum_i \int_s p(i,s) \log\left[\frac{p(i,s)}{p(i)p(s)}\right]$$

From bayes rule, $p(i,s) = p(s|i)p(i)$we get

$$I(i;s) = \sum_i \int_s p(s|i)p(i) \log\left[\frac{p(s|i)p(i)}{p(i)p(s)}\right]$$

Cancelling and factoring $p(i)$

$$I(i;s) = \sum_i p(i) \int_s p(s|i) \log\left[\frac{p(s|i)}{p(s)}\right]$$

The second term is the KL-Divergence between $p(s|i)$and $p(s)$, so we’re measuring the average KLD between a team’s skills, and the overall distribution of skills on the server. Because teams are constrained to have the same number of players, p(i) is constrained to be 0.5 for both teams.

If $p(s|i)$and $p(s)$are both normal with params μ_i, σ_i and μ, σrespectively, then this becomes (cribbed from wikipedia for the KLD between two normals.

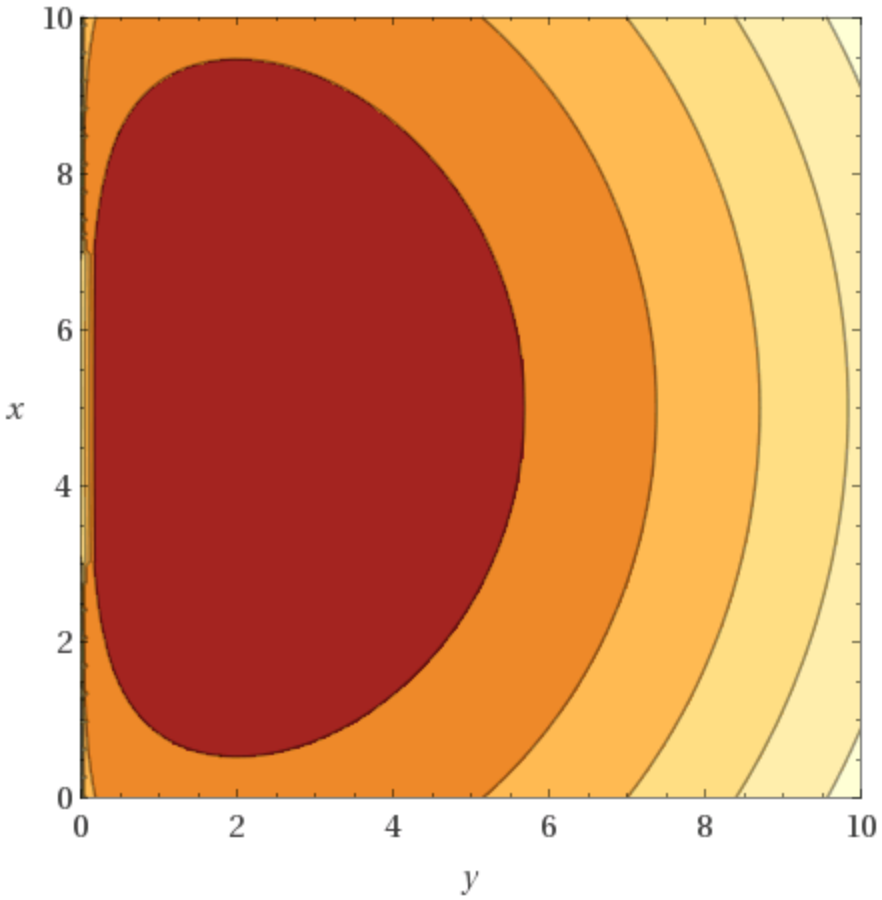
$$\sum_i \frac{1}{2}\left[\frac{(\mu_i-\mu)^2}{\sigma^2} + \frac{\sigma_i^2}{\sigma^2} - \log\frac{\sigma_i^2}{\sigma^2} - 1\right]$$

The goal of balancing the teams is to make this metric as small as possible, that is, each team should have a distribution of skills that is as close as possible to the overall distribution of skills on the server. In this case, p(i) is constrained to be 0.5 for both teams, because we want the teams to have the same number of players, so we can ignore it.

As a practical matter, $\log\frac{\sigma_i^2}{\sigma^2}$goes to -∞ if the standard deviation is 0 (if all of the players have the same skill) and the server’s standard deviation is in the denominator throughout, so we need to

smooth our estimates of the standard deviations by adding a constant prior to our standard deviation calculation. This roughly corresponds to assuming that each player's skill isn't a fixed quantity but has some small distribution of values.

Example plot:



The gradients are $\sum_i \frac{\mu_i - \mu}{\sigma^2}$ and $\sum_i \left[\frac{\sigma_i}{\sigma^2} - \frac{1}{\sigma_i}\right]$ which make the criteria for minimizing this a little more clear.

We can incorporate team preference into this using the same strategy. Define a bernoulli variable for each player that describes their team preference from 0 to 1. We want to *maximize* the mutual information between the team preference and the team selection. This ensures that players who prefer different teams are placed on different teams. It does not specify which team should be which, so the teams should be swapped after the optimization so that as many of the team preferences as possible match up.. Let t_i be the average team preference of team i and t be the average over the whole server. The mutual information is then proportional to

$$\sum_i t_i \log\left(\frac{t_i}{t}\right) + (1 - t_i)\log\left(\frac{(1-t_i)}{1-t}\right)$$

So in total, we want to minimize

$$\sum_i \frac{1}{2}\left[\frac{(\mu_i-\mu)^2}{\sigma^2} + \frac{\sigma_i^2}{\sigma^2} - \log\frac{\sigma_i^2}{\sigma^2} - 1\right] - \alpha\sum_i t_i \log\left(\frac{t_i}{t}\right) + (1 - t_i)\log\left(\frac{(1-t}{1-}\right.$$

where $\alpha$is a parameter to control how much we care about team preference.

We can optimize this either by hill climbing, or simulated annealing.