Bittensor: A Peer-to-Peer Intelligence Market

Constantine Damore

www.bittensor.com

Abstract

Like with other commodities, markets could help us efficiently produce machine intelligence. We propose a market where intelligence is priced by other intelligence systems peer-to-peer across the internet. Peers rank each other by training neural networks which learn the value of their neighbors. Scores accumulate on a digital ledger where high ranking peers are monetarily rewarded with additional weight in the network. However, this form of peer-ranking is not resistant to collusion which disrupts the accuracy of the mechanism. The solution is a connectivity based regularization which exponentially rewards trusted peers making the system resistant to collusion of up to 50 percent of the network weight. The result is a collectively run intelligence market which pays contributors who create information theoretic value and continual produces newly trained models.

The production of machine intelligence has come to rely almost entirely on a system of benchmarking where machine learning models are trained to perform well on narrowly defined supervised problems. While this system works well for pushing the performance on these specific problems, the mechanism suffers where markets could excel. For one, intelligence is increasingly becoming un-tethered from specific objectives to become a commodity which is, expensively mined from data [1], monetarily valuable [2], transferable [3], and generally useful [4]. Measuring its production with supervised objectives does not directly reward the commodity itself and converges the field towards narrow specialists [5]. Moreover, because these objectives (often measured in singular metrics like accuracy) do not have the resolution to reward niche or legacy systems, what is not currently state of the art is lost. Ultimately, the proliferation of diverse intelligence systems is limited by the need to train large monolithic models to succeed in a winner take all competition. Standalone engineers cannot directly monetize their work and what results is centralization where a small set of large corporations control access to the best artificial intelligence [2].

A new commodity is in need of a new form of market[1]. This paper suggest a framework in which machine intelligence is measured by other intelligence systems. Models are ranked for informational production regardless of the subjective task or dataset used to train them. By changing the basis against which machine intelligence is measured, the market can reward intelligence general to a much larger set of objectives, legacy systems can be monetized for their unique value, and smaller diverse systems can find niches within a much higher resolution reward landscape. The solution is a network of computers who share representations with each other in a continuous and asynchronous fashion, peer-to-peer (P2P) across the internet. The constructed market uses a digital ledger to record ranks and provide incentive to the peers in a decentralized manner. The chain measures trust, making it difficult for peers to attain reward without providing value to the majority. Researchers can directly monetize machine intelligence work and consumers can directly purchase it.

[1]"The iron rule of nature is: you get what you reward for. If you want ants to come, you put sugar on the floor." - Charlie Munger

Preprint. Under review.

1 Model

We begin with an abstract definition of intelligence [6] in the form of a parameterized function $y = f(x)$ trained over a dataset $D = [X, Y]$ to minimize a loss $\mathcal{L} = E_D[\mathcal{Q}(\ y,\ f(x))\ )]$. Our network is composed of n functions $F = f_0, ..., f_j, ...f_n$, 'peers' where each is holding zero or more network weight $\mathbf{S} = [s_i]$ 'stake' represented on a digital ledger. These functions, together with losses and their proportion of stake, represent a stake-weighted machine learning objective $\sum_i^n \mathcal{L}_i * s_i$.

$$\begin{array}{cccccc}
\mathcal{L}_0 & \mathcal{L}_1 & \mathcal{L}_2 & \mathcal{L}_3 & \mathcal{L}_4 & \mathcal{L}_5 \\
\uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow \\
f_0 & f_1 & f_2 & f_3 & f_4 & f_5 \\
\uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow \\
D_0 & D_1 & D_2 & D_3 & D_4 & D_5
\end{array}$$

Figure 1: Peer functions with losses $\mathcal{L}_i$ and unique datasets D_i.

Our goal is the distribution of stake to peers who have helped minimize this loss-objective Figure-1. Our proposal achieves this through *peer-ranking*, where peers use the outputs of others $F(x) = [f_0(x)...f_n(x)]$ as inputs to themselves $f(F(x))$ and learn a set of weights $\mathbf{W} = [w_{i,j}]$ where each $w_{i,j}$ is the locally calculated score attributed to the j^{th} peer from the i^{th}.

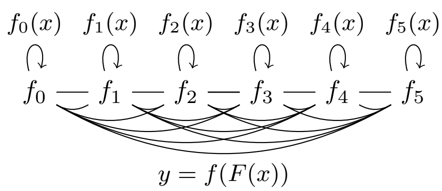


Figure 2: Inter-function connectivity.

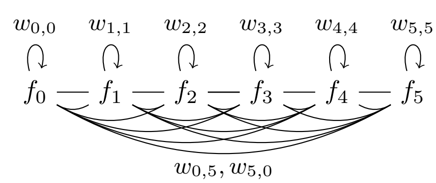


Figure 3: Weight matrix.

Peers submit weight-updates to the digital ledger in the form of transactions which fill the blocks appended to the chain. Updates at each blockstep t are then applied using a common learning rate λ, $\mathbf{W}_{t+1} = \mathbf{W}_t + \lambda \Delta \mathbf{W}$. Using these weights, the digital ledger can compute ranks $\mathbf{R} = [r_i]$, the overall-score for each peer in the network and proportionally inflate newly minted stake $\Delta \mathbf{S}_t = [\Delta s_i]$ to those peers. As inflation progresses at a rate τ, peers with significance to the overall objective are rewarded more and come to own a larger proportion of the network.

$$\mathbf{R} = \mathbf{W}^T \cdot \mathbf{S} \tag{1}$$

$$\mathbf{S}_{t+1} = \mathbf{S}_t + \tau \cdot \frac{\mathbf{R}}{||\mathbf{R}||} ||\mathbf{S}|| \tag{2}$$

2 Computing Weights

Peers learn weights $w_{i,j}$ by computing the value of other peers towards their own loss function $\mathcal{L}$. For instance, a suitable score for the j^{th} peer from the i^{th} is the change to $\mathcal{L}_i$ when that peer is removed from the network [7]. Representing this removal as an input perturbation $\Delta F(x)_j = [0, ...0, -f_j(x), 0, ...0]$ this score can be derive as follows (Appendix 13.1):

$$w_{i,j} = \sum_{x \in D_i} \Delta F^T(x)_j \cdot \mathbf{H}(\mathcal{Q}_i(x)) \cdot \Delta F(x)_j \tag{3}$$

Peers are not required to compute scores in this manner, for instance, it may be computationally expensive to compute the hessian term $\mathbf{H}$ and more efficient to use a heuristic to propagate a score from the loss function through to the inputs [8]. However, where $\mathcal{Q}$ is the common cross-entropy loss, the hessian term $\mathbf{H}(\mathcal{Q}_j)$ is the Fisher-information matrix [8], and the ranks produced by Equation (3) are the stake weighted *informational significance* of each peer to the network as a whole.

The immediate problem with this stake-incentive system is that the computation of $w_{i,j}$ in Equation (3) is non-auditable. It is not possible to enforce that weights are honestly reported without access to the parameters of each function, information we do not have on the distributed ledger. Instead, it is reasonable to assume most, if not all, participants will select weights which artificially increase their own rank – undermining accuracy in the market.

3 Rewarding Trust

Our solution begins with a convention: messages from peer i to peer j are queued w.r.t to the stake weight $w_{i,j} * s_{i,j}$ between them. Peers are persuaded to increase weights towards a game-theoretic equilibrium to better access the network (Appendix 13). However, competition provides only part of the solution: peers with little competitive interest in attaining value from their neighbors may still collude to gain inflation without adding value to the network. This is achieved by forming a 'cabal', a set of one-or-more tightly connected peers who falsely evaluate each other. The default incentive mechanism rewards this behaviour with inflation proportional the total stake held by the sub-graph.

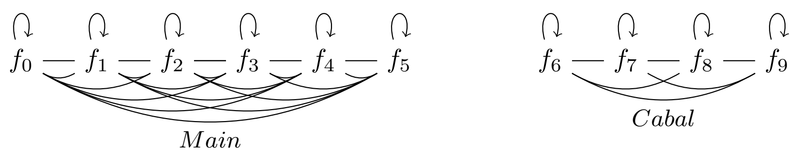


Figure 4: Disjoint cabal.

We disincentivize the formation of cabals by promoting connectivity within the graph. Peers must be tightly nested within the largest sub-graph to attain the highest proportion of inflation while smaller disjoint sub-graphs decay overtime. We use an absorbing markov chain calculation to compute a matrix $\mathbf{C}$ where $c_{i,j} = s_j$ if and only if there is a greater than some threshold likelihood of visiting peer j from i. Computing likelihoods is done with respect to the transition probabilities $w_{ij} \in \mathbf{W}$ and the threshold for connectivity can be set in the chain configuration. We then update the ranking equation 1 like so:

$$\mathbf{R} = (\mathbf{W} + \mathbf{C})^T \mathbf{S} \tag{4}$$

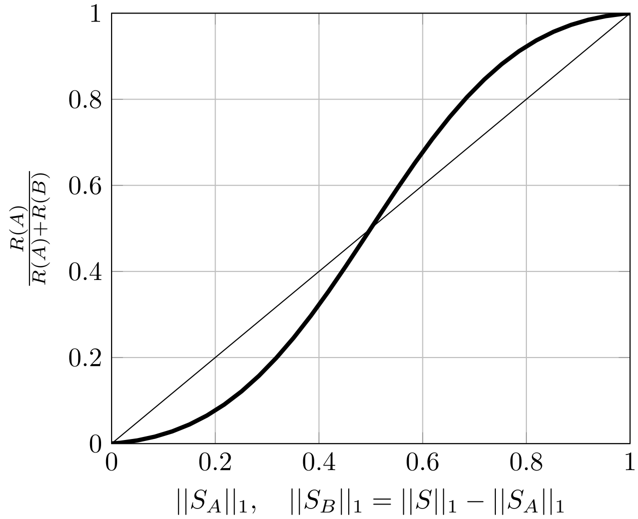
Note that for a tightly connected graph, $c_{i,j} = s_j$ for all $c_{i,j} \in C$ and $||\mathbf{C}^T \mathbf{S}||_1 = ||\mathbf{S}||_1^2$. Further more, the weight matrix $\mathbf{W}$ is row normalized such that $||\mathbf{W}^T \mathbf{S}||_1 = ||\mathbf{S}||_1$ and we have:

$$||\mathbf{R}||_1 = ||(\mathbf{W} + \mathbf{C})^T \mathbf{S}||_1 = ||\mathbf{S}||_1 + ||\mathbf{S}||_1^2 \tag{5}$$

The extra regularization term $||\mathbf{S}||_1^2$ ensures that the quantity of inflated stake in a tightly connected graph is quadratically related to its size. By normalizing $\mathbf{S}$ to a fixed size $||S||_1 = \sum_{s_i \in \mathbf{S}} s_i = \rho$ we can fix this effect for sub-graphs based on proportion of held stake.

$$\sum_{r_i \in R} r_i = ||(\mathbf{W} + \mathbf{C})^T \cdot \mathbf{S}||_1 = \rho + \rho^2 \tag{6}$$

Ratio of inflation between two competing sub-graphs A and B with $||S||_1 = \rho = 10$



We can see that with continuous inflation the sub-graph with greater than 50 percent of stake will dominate the network over time.

4 Annealing Trust

The network is only resistant to a disjoint sub-graph with up to 50 percent of the network stake if there is a larger connected component in the graph, notably with >50 percent. Resistance to an attack of this size can be fixed by noting that the sum of ranks $||\mathbf{R}||_1$ will exceed $2 \cdot (\frac{\rho}{2} + \frac{\rho}{2}^2)$ if and only if there is a connected component larger than 50 percent. The weight matrix $\mathbf{W}$ is row normalized and so this depends entirely on the matrix $\mathbf{C}$. Each peer is responsible for a score of exactly $s_i \cdot ||\mathbf{S}||_1$ in the computation of $||\mathbf{R}||$ (7), if that peer is outwardly dis-trusting (i.e. not connected to other peers with high confidence, $c_{i,j} = 0$ for all j, and $\mathbf{R}$ loses additional weight equivalent to $s_i \cdot ||\mathbf{S}||_1$.

$$||\mathbf{C}^T\mathbf{S}|| = \begin{bmatrix} s_0 & s_0 & s_0 & s_0 \\ s_1 & s_1 & s_1 & s_1 \\ s_2 & s_2 & s_2 & s_2 \\ s_3 & s_3 & s_3 & s_3 \end{bmatrix} \begin{bmatrix} s_0 \\ s_2 \\ s_3 \\ s_4 \end{bmatrix} = \begin{bmatrix} s_0 \cdot (s_0 + s_1 + s_2 + s_3) \\ s_1 \cdot (s_0 + s_1 + s_2 + s_3) \\ s_2 \cdot (s_0 + s_1 + s_2 + s_3) \\ s_3 \cdot (s_0 + s_1 + s_2 + s_3) \end{bmatrix} = \sum_i s_i \cdot (s_0 + s_1 + s_2 + s_3) \tag{7}$$

To ensure sufficient outward-trust we provide incentive for peers to set weights such that $\sum_j c_{i,j} = \frac{1}{2}||\mathbf{S}||_1$. At this point this peer is outwardly connected to greater than 50 percent of the network. If all peers meet this requirement then $||\mathbf{C}^T S|| > 2 \cdot ||\frac{\mathbf{S}}{2}||_1^2 = 2 \cdot \frac{\rho}{2}^2$ and we meet our resistance level of $2 \cdot (\frac{\rho}{2} + \frac{\rho}{2}^2)$. If peers don't meet this requirement we linearly clip their stake weight in the graph as below.

$$s_i^* = s_i \cdot \min(\frac{\sum_j c_{i,j}}{\frac{1}{2}||\mathbf{S}||_1}, 1) \tag{8}$$

Stake clipping provides incentive for peers to reach the connection requirement of $\sum_j c_{i,j} = \frac{1}{2}||\mathbf{S}||_1$, if they fail to meet this value, $s_i^* < s_i$ the peer losses weight in the inflation mechanism. While it is not a guaranteed that peers reach this connection requirement, 8 is differentiable in the weights and peers can use gradient descent to maximize their stake weight in conjunction with computing their weights normally. If peers fail to meet this requirement we are still guaranteed that $||\mathbf{R}||_1$ exceeds our resistance point when measured across post-clipped stake values

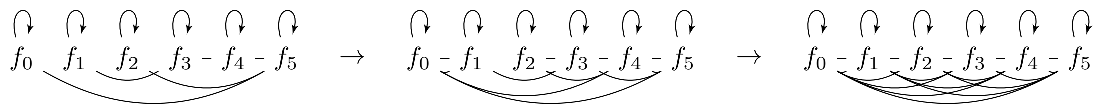


Figure 5: Peers with incentive to maximize outward trust.

5 Transactions and Token Inflation

A single token can be used to bootstrap the entire market. Inflation past this point is triggered by transactions on chain. Transaction fees can be made in the same staking token keeping the incentives self-contained within the network. Peers with no initial stake can still connect to the network by advertising their existence and having peers already online make the initial transactions to subscribe them to the weight matrix.

The proportion of inflation to emit e_t at each transaction is determined by the last token emission Δt from that peer. The chain need only remember when the last inflation step occurred. Direct neighbors attain the designated emission value from $\mathbf{W}$ and scores from the matrix $\mathbf{C}$ are computed using a depth first recursion without the need for expensive matrix inverse calculations.

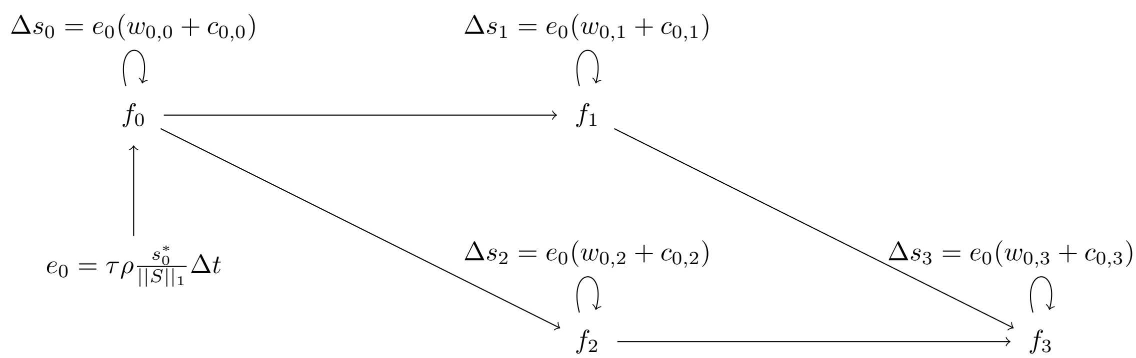
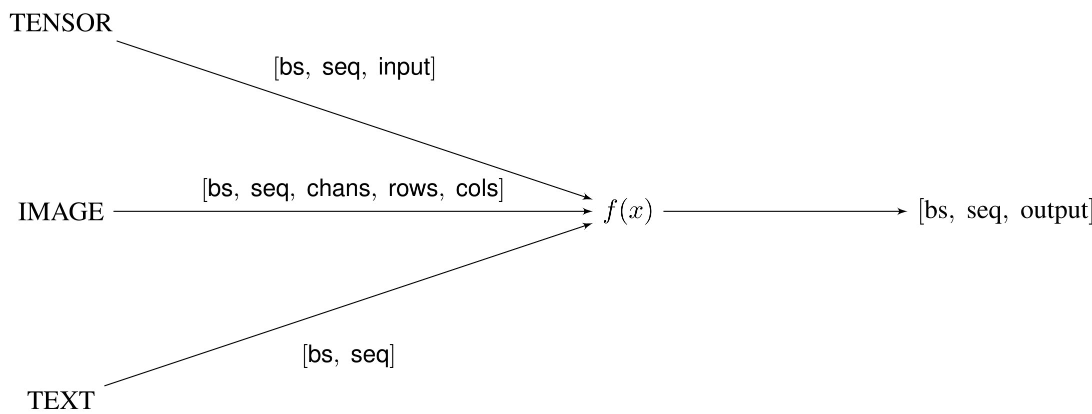


Figure 6: Emission of new stake.

The self-loop $w_{i,i}$ provides the peer with its share of self-triggered inflation, because of this there is incentive to make regular updates to the ledger. Peer which make more transactions gain marginally more because of the compounding inflation measured on more timesteps. The need to continually submit transactions means the largest sub-graph must perform work (monetary expense in the form of transactions) to continually maintain its dominance in the graph. Similar to Bitcoin this limits the potential for long term control-attacks on the network, only those that continue to train the weight matrix are rewarded with the highest proportion of inflation.

6 Tensor Standardization

A common encoding of inputs and outputs is required for various model types and input types to interact. We use a standard output shape across the network [batch_size, sequence_dim, output_dim] – similar to the common tensor-shapes produced by language and image models. Inputs types are passed within TEXT, IMAGE, TENSOR modalities and an additional sequence dimension is inserted to extend the network into the temporal domain.



The abstract scope of inputs ensures participants can be multi-task [9], use completely distinct computing substrates [10] or train on unique datasets [11].

7 Conditional Computation

As the network grows, outward bandwidth is likely to become the major bottleneck and a method of finding peers valuable to each model is required. The solution is to use conditional computation at the peer level where peers learn using a product key layer or a sparsely gated layer [12] to choose which peers to query for each example.

$$f_i = f_i(G(x)) \tag{9}$$

$$G(x) = \sum_j g_j(x) * f_j(x) \tag{10}$$

The conditional layer determines a sparse combination of peers to query for each example and multiplicatively re-joins them, cutting outward bandwidth by querying only a small subset of peers for each example. The method drastically increase the potential for outward bandwidth [12] [13] allowing peers to communicate with many more neighbors in the graph. In essence the layer acts as a trainable DNS lookup for peers based on inputs and being trainable w.r.t to the loss provides a useful proxy for the importance scores $w_{i,j} \in \mathbf{W}$.

8 Knowledge Extraction

Dependence between functions ensures that models must stay online and cannot be run in production. Breaking this dependence can be achieved using distillation[6]: a compression and knowledge extraction technique in which a smaller model – the student - mimics the behaviour of the remaining network. The distillation layer is employed in conjunction with conditional computation (10) layer where the student model learns to mimic the network using the cross-entropy (shown below as KL) between the logits produced by the gating network and the student's predicted distribution [14].

$$\text{distillation loss} = \text{KL}_D(\text{dist}(x), G(x)) \tag{11}$$

Because the distilled model acts as a proxy for the network, models can be fully taken off-line and evaluated. Recursion through the network is also cut between components allowing for arbitrary network graphs. If models go offline, their peers can use the distilled versions in-place. Private data can be validated over the distilled models instead of querying the network. Eventually, components can fully disconnect from the network using the distilled inputs to validate and inference the models offline.

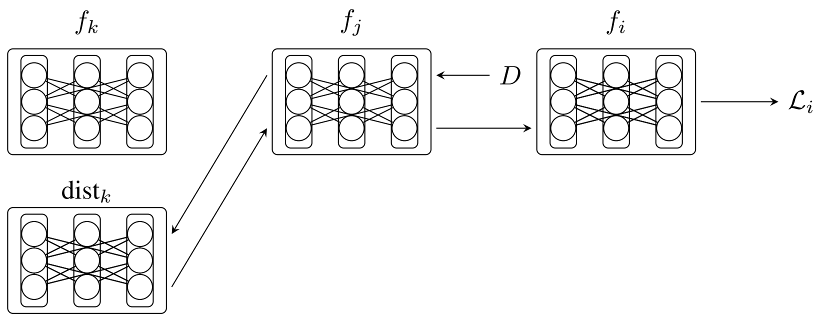


Figure 7: Queries propagate to depth=1 before the distilled model is used.

9 Running the Network

The steps to run a peer in the network are:

1. The peer defines its dataset D_i, loss $\mathcal{L}_i$ and parameterized function f_i
2. At each training iteration the peer conditionally broadcasts batches of examples from D_i to its peers $x = $ [batch_size, sequence_length, input_size]
3. The responses $F(x) = [...f_j(x)...]$ – each of common shape $f_j(x) =$ [batch_size, sequence_length, output_size] – are joined using the gating function and used as input to the local model f_i.
4. Comparison against the target labels produces a loss-gradient $\frac{\partial \mathcal{L}}{\partial F}$ which back-propagates through f_i and out to the network.
5. During 2 and 3 the peers learn the weights for their row $w_{i,j} \in \mathbf{W}$ by measuring the value of the signals produced by their peers.
6. At distinct time-steps t participants submit changes to the weights $\Delta \mathbf{W}_i$ to update the ranking $\mathbf{R}$.
7. $\mathbf{R}$ translates into newly minted stake ΔS distributed on the digital ledger.

10 Collusion

We consider the scenario where a subset of the nodes in the network have formed a 'cabal' a set of colluding nodes attempting to maximize their inflation without accurately scoring their neighbors. The fight between the honest graph A with stake $\mathbf{S}_A$ and the disjoint cabal B with stake $\mathbf{S}_B$ is determined by the proportion of inflation attained by each, the honest graph must attain proportionally more inflation to maintain its dominance.

Given our connection based incentive (5) in the worst case, the cabal forms a κ-graph structure seen in Figure-(4) with weights set equally between each member. Since this sub-graph is well connected $c_{i,j} = s_j$ for all $c_{i,j} \in \mathbf{C}_B$ and the sum of ranks in sub-graph B is given by:

$$\sum_{r_b \in B} r_b = ||(\mathbf{W} + \mathbf{C})^T \cdot \mathbf{S}_B||_1 = \frac{\rho ||\mathbf{S}_B||_1}{||\mathbf{S}||_1} + \frac{(\rho ||\mathbf{S}_B||_1)^2}{||\mathbf{S}||_1} \tag{12}$$

The sum of ranks $||R||_1$ exceeds or equal our pegged value $2 \cdot (\frac{\rho}{2} + \frac{\rho}{2}^2)$ when measuered across stake which has been post-scaled according to our method in Section-4. This cannot occur unless there is a connected component holding more than 50 percent of the network stake. Since the cabal holds less than 50 percent, the honest graph A must be tightly connected. Trivially, since $\mathbf{S}_A > \mathbf{S}_B$, the inflation in the honest graph $||\mathbf{S}_A||_1 + ||\mathbf{S}_A||_1^2$ exceeds the magnitude of inflation in the disjoint cabal $||\mathbf{S}_B||_1 + ||\mathbf{S}_B||_1^2$. We compute the relative size of the disjoint graph as a function of steps below.

```
import math
def cabal_decay(prcnt_c: float, inf_rate: float, rho: float, n_steps: int):
```

```
    stake = 1
    for step in range(n_steps):
        prcnt_g = (1 - prcnt_c)
        R_c = prcnt_c + rho * prcnt_c * prcnt_c
        R_g = prcnt_g + rho * prcnt_g * prcnt_g
        infl_c = R_c / (R_c + R_g)
        infl_g = R_g / (R_c + R_g)
        stake_c = (stake * prcnt_c) + (inf_rate * stake * infl_c)
        stake_g = (stake * prcnt_g) + (inf_rate * stake * infl_g)
        stake = stake_c + stake_g
        prcnt_c = stake_c / stake
        print (step, prcnt_c)

>> cabal_decay (prcnt_c = 0.49, inf_rate = 0.1, rho = 100, n_steps = 50 )
0 0.4891094401482484
1 0.4881397135369698
2 0.48708382424553454
3 0.4859341702598143
4 0.4846824945044336
5 0.48331983296592057
6 0.48183646006207687
7 0.4802218315423974
8 0.4784645253763857
9 0.4765521813125239
10 0.4744714400843485
...
40 0.24535389385917838
41 0.23195275928239806
42 0.218670910170184
43 0.20559383494222536
44 0.19280020696003405
45 0.18035995913333952
46 0.1683329646952147
47 0.1567683325477391
48 0.14570426763658745
49 0.13516840696778115
```

11 Conclusion

We have proposed a intelligence market which runs on a P2P network setting outside of a trusted environment. Crucially, the benchmark measures performance as representational knowledge production using other intelligence systems to determine its value. The fact that this can be done in a collaborative and high resolution manner suggests the benchmark could provide a better reward mechanism for the field in general. To achieve this aim the paper began with the definition of P2P network composed of abstractly defined intelligence models. We showed how this framework allowed us to produce a ranking for each peer based on the cost to prune it from the network. Peers negotiated this score using a set of weights on a digital ledger. However, the system was incomplete without mechanisms that prevented participants from forming dishonest sub-graphs. To resolve this we proposed an incentive scheme based on peer connectivity which exponentially rewarded peers for being trusted by a large portion of the network, this ensured that over time dishonest sub-graphs decay to irrelevance. Following this, we showed how peers reduced the network bandwidth by learning connectivity using a differential layer and how they could extract fully network-disconnected machine learning models to run in production. The result is an intelligence market which rewards participants for producing knowledge and making it available to new learners in the system.

References

[1] R. Schwartz, J. Dodge, N. A. Smith, and O. Etzioni, "Green ai," 2019.

[2] OpenAI, "Openai licenses gpt-3 technology to microsoft," *OpenAI Blog*, vol. 1, no. 1, p. 1, 2020.

[3] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[4] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI Blog*, vol. 1, no. 8, p. 9, 2019.

[5] F. Chollet, "On the measure of intelligence," *arXiv preprint arXiv:/1911.01547*, 2019.

[6] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[7] Y. LeCun, D. J. S, and S. S. A, "Optimal brain damage," *Advances in Neural Information Processing Systems 2 (NIPS)*, 1989.

[8] R. Yu, A. Li, C.-F. Chen, J.-H. Lai, V. I. Morariu, X. Han, M. Gao, C.-Y. Lin, and L. S. Davis, "Nisp: Pruning networks using neuron importance score propagation," 2017.

[9] L. Kaiser, A. N. Gomez, N. Shazeer, A. Vaswani, N. Parmar, L. Jones, and J. Uszkoreit, "One model to learn them all," 2017.

[10] M. A. Nugent and T. W. Molter, "Cortical processing with thermodynamic-ram," 2014.

[11] G. Lample and A. Conneau, "Cross-lingual language model pretraining," 2019.

[12] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean, "Outrageously large neural networks: The sparsely-gated mixture-of-experts layer," 2017.

[13] M. Riabinin and A. Gusev, "Learning@home: Crowdsourced training of large neural networks using decentralized mixture-of-experts," *arXiv preprint arXiv:/2002.04013*, 2020.

[14] L. C. J. W. T. Sanh, Victor; Debut, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter," *arXiv preprint arXiv:1910.01108*, 2019.

[15] D. Balduzzi, W. M. Czarnecki, T. W. Anthony, I. M. Gemp, E. Hughes, J. Z. Leibo, G. Piliouras, and T. Graepel, "Smooth markets: A basic mechanism for organizing gradient-based learners," 2020.

[16] P. Dütting, Z. Feng, H. Narasimhan, D. C. Parkes, and S. S. Ravindranath, "Optimal auctions through deep learning," 2017.

article

12 Appendix

13 Analysis

We consider the accuracy of the ranking mechanism where peers make self interested updates to the weights on chain. To do this we model each peer's payoff function in two terms.

$$P_i(\mathbf{W}) = U_i(\mathcal{L}_i(\mathbf{W})) + r(\mathbf{W})_i \tag{13}$$

1. Peer i's rank $r_i(\mathbf{W})$ as a function of the weights.
2. A utility term attached to peer i's loss as a function of the weights $U(\mathcal{L}(\mathbf{W}))$.

The change in inputs induced by a change in the weights can be modelled using a threshold function, in our case a shifted sigmoid function, where inputs from neighbors are masked when weights drop bellow the average set by other peers $\mu_j = (\frac{1}{n}) \sum_i^n s_i * w_{i,j}$

$$F_{\mathbf{W}}(x) = [f_0(x) * \sigma(s_i * w_{i,0} - \mu_0), ..., f_n(x) * \sigma(s_i * w_{i,n} - \mu_n)] \tag{14}$$

$$\sigma = \frac{1}{1 + e^{-\frac{x}{T}}} \tag{15}$$

We then derive the change in loss given a change in weights through an input perturbation $(F_{\mathbf{W}} - F_{\mathbf{W}^0})$ where $\mathbf{W}^0$ is the initial choice of weights. Using the same perturbation equation from Section 1 we can then reflect the change in loss using a simulated Hessian term $\mathbf{H}(\mathcal{L}(F))$ as a function of the weights $\frac{\partial L}{\partial \mathbf{W}}$ (see 13.2):

$$\frac{\partial L}{\partial \mathbf{W}} = \frac{\partial}{\partial \mathbf{W}} \left[(F_{\mathbf{W}} - F_{\mathbf{W}^0})^T \cdot H(\mathcal{L}(F)) \cdot (F_{\mathbf{W}} - F_{\mathbf{W}^0}) \right] \tag{16}$$

We make a further linear assumption about the utility function $U_i(W) = \alpha \cdot \mathcal{L}_i(W)$ to give us a fully differential function for a peer's utility. This construction is a smooth market [15] where we can explore the competitive equilibrium using gradient descent [2] with steps $\Delta \mathbf{W}_i = \frac{\partial P_i}{\partial \mathbf{W}_i}$.

$$\mathbf{W}_{t+1} = \mathbf{W}_t + \lambda \Delta \mathbf{W} \tag{17}$$

$$\Delta \mathbf{W} = \left[\frac{\partial P_0}{\partial \mathbf{w}_0}; \cdots ; \frac{\partial P_n}{\partial \mathbf{w}_n} \right] \tag{18}$$

To evaluate the the accuracy of the peer ranking method, we generate statistics from the above empirical model. We first select mechanism parameters $[\rho, \lambda, \alpha, n]$ and generate an initial randomized network state $[\mathbf{W}^0, \mathbf{S}]$ and n random positive semi-definite $n \times n$ hessian terms $[\mathbf{H}]$ one for each peer. Given the initialization we apply the descent strategy (18) by computing the gradient terms from (16) and converge the system to the implied equilibrium using a standard gradient descent toolkit. The discovered local minimum is the competitive equilibrium where participants cannot vary from their choice of weights and stand to gain [16]. At this point we compute the competitive ranking $\mathbf{R}^*$ and compare it to the idealized score $\mathbf{R}$ derived from the hessians and discusses in Section 2. We measure the difference between the two scores as a spearman-rho correlation and plot example trials bellow.

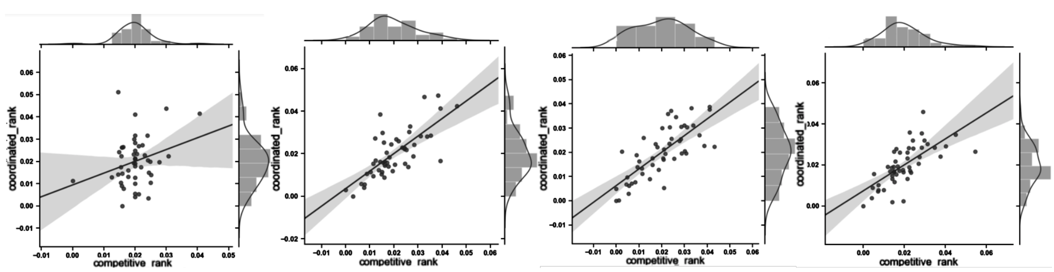


Figure 8: Correlations between the competitive rank and coordinated rank for $\alpha \in \{1, 10, 25, 50\}$. We note that we see an increased relationship between the idealized rank and those discovered by the market improves increasingly through the parameter α.

13.1 Deriving the idealized ranking

We approximate the change in the benchmark $\mathcal{B} = \sum_i^n \mathcal{L}_i$ at a local minimum and under a perturbation $\Delta F(x)_i = [..., -f_i(x), ...]$ reflecting the removal of the i^{th} node.

$$\Delta \mathcal{B} = \mathcal{B}(F + \Delta F_i) - \mathcal{B}(F) = \sum_i^n \mathcal{L}_i(F + \Delta F_i) - \mathcal{L}_i(F) \tag{19}$$

$$\mathcal{L}_i(F + \Delta F_i) - \mathcal{L}_i(F) \approx \frac{\partial \mathcal{L}_i}{\partial F} \cdot \Delta F_i + \frac{1}{2} \Delta F_i^T \cdot H(\mathcal{L}_i) \cdot \Delta F_i + O(\Delta F_i^3) + O(\Delta F_i^3) \tag{20}$$

Equation (19) follows from the definition of the benchmark and Equation (20) follows from a Taylor series under the perturbation $\Delta F(x)_i$. Note that the first term $\frac{\partial \mathcal{L}_i}{\partial F}$ is zero at the local minimum and the higher order term

[2] Making gradient steps in this game is a regret-free strategy (see 13.5) and achieves the best expected payoff in hindsight.

$O(\Delta F_i^3)$ can be ignored for sufficiently small perturbations. These assumptions are also made by [7] and [8]. Note that $\mathcal{L}_i$ is an expectation over the dataset D_i, and all terms are evaluated at a point x so we have:

$$\Delta \mathcal{B} \approx \frac{1}{2} \sum_{i}^{n} \sum_{x \in D_i} \Delta F_i^T(x) \cdot H(\mathcal{Q}_i(x)) \cdot \Delta F_i(x) \tag{21}$$

Here the hessian over the error function $H(\mathcal{Q}_i(x))$ and the summation over the dataset $\sum_{x \in D_i}$ have been appropriately substituted. The constant factor $\frac{1}{2}$ can be removed and this leaves our result.

13.2 Deriving the weight convergence game.

13.3 Theorem

For choice of Hessians $H(\mathcal{L}(F))$ the network convergence-game can be described with the following linear relationship between gradient terms:

$$\frac{\partial P}{\partial \mathbf{W}} = \alpha \cdot \frac{\partial L}{\partial \mathbf{W}} + \frac{\partial r}{\partial \mathbf{W}} \tag{22}$$

With the gradient of the loss:

$$\frac{\partial L}{\partial \mathbf{W}} = \frac{\partial}{\partial \mathbf{W}} [(F_{\mathbf{W}} - F_{\mathbf{W}_0})^T \cdot H(\mathcal{L}(F)) \cdot (F_{\mathbf{W}} - F_{\mathbf{W}_0})] \tag{23}$$

13.4 Setup

We analyze the system by characterizing the behaviour of participants via their payoff in two terms:

1. The utility attached to that participant's loss as a function of their weights is $U(L(\mathbf{W}))$. U is assumed roughly linear for small change in the weight matrix, $U(\mathcal{L}) = \alpha * \mathcal{L}$, with $\frac{\partial U}{\partial \mathcal{L}} = \alpha$, and α is assumed positive and non-zero.
2. The network is converged to a local minimum in the inputs $\frac{\partial \mathcal{L}}{\partial F} = 0$.

From the payoff formulation in 6.3 we write:

$$P(\mathbf{W}) = \alpha \cdot \mathcal{L}(\mathbf{W}) + r(\mathbf{W}) \tag{24}$$

Note, the utility function and emission were measured in similar units and so α is the *price* of each unit change in loss. The analysis just supposes such a score exists, not that it can be computed. Participants are selecting their weights by making gradient steps $\Delta \mathbf{W}_i = \frac{\partial P_i}{\partial \mathbf{W}_i}$ as to maximize their local payoff. For brevity we omit the subscript i for the remainder of the analysis. Consider a Taylor expansion of the loss under a change ΔF in the inputs.

$$\mathcal{L}(F + \Delta F) = L(F) + \frac{\partial \mathcal{L}}{\partial F} \Delta F + \frac{1}{2} \Delta F \cdot H(\mathcal{L}(F)) \cdot \Delta F + O(\Delta F^3) \tag{25}$$

The first linear term $\frac{\partial L}{\partial F}$ is zero at the assumed minimum and the higher order terms are removed for sufficiently small perturbations in F. We then perform a change of variable $F = F_{\mathbf{W}_0}$, and $\Delta F = F_{\mathbf{W}_1} - F_{\mathbf{W}_0}$ where $\mathbf{W}_0$ are the weights at the minimume and $\mathbf{W}_1$ are another choice such that $F_{\mathbf{W}_0}$ and $F_{\mathbf{W}_1}$ are those inputs masked by $\mathbf{W}_0$ and $\mathbf{W}_1$ accordingly. Substituting this into Equation (31):

$$\mathcal{L}(F_{\mathbf{W}_1}) = L(F_{\mathbf{W}_0}) + \frac{1}{2} (F_{\mathbf{W}_1} - F_{\mathbf{W}_0})^T \cdot H(\mathcal{L}(F)) \cdot (F_{\mathbf{W}_1} - F_{\mathbf{W}_0}) \tag{26}$$

The function $\mathcal{L}(F_{\mathbf{W}_1})$ is simply an approximation of the loss for any choice of weights $\mathbf{W}_1$ given that the network has already converged under $\mathbf{W}_0$. Finally, by the α-linear assumption of the utility we can attain the following:

$$\frac{\partial U}{\partial \mathbf{W}} = \alpha \cdot \frac{\partial L}{\partial \mathbf{W}} \approx \frac{\alpha}{2} \frac{\partial}{\partial \mathbf{W}} [(F_{\mathbf{W}} - F_{W_0})^T \cdot H(\mathcal{L}(F)) \cdot (F_{\mathbf{W}} - F_{W_0})] \tag{27}$$

Note that we've dropped the subscript $\mathbf{W}_1$ for brevity, $L(F_{\mathbf{W}_0}))$ is constant and therefore not depending on the choice of weights, and the fraction $\frac{1}{2}$ can be safely subsumed into the unknown α. The remaining term $\frac{\partial r}{\partial \mathbf{W}}$ is derivable via the ranking ranking function in Section 1.2. This leaves the result:

$$\frac{\partial P}{\partial \mathbf{W}} \approx \alpha \cdot \frac{\partial L}{\partial \mathbf{W}} + \frac{\partial R}{\partial \mathbf{W}} \tag{28}$$

$$\frac{\partial L}{\partial \mathbf{W}} = \frac{\partial}{\partial \mathbf{W}} [(F_{\mathbf{W}} - F_{\mathbf{W}_0})^T \cdot H(\mathcal{L}(F)) \cdot (F_{\mathbf{W}} - F_{\mathbf{W}_0})] \tag{29}$$

13.5 Deriving the ex-post zero-regret step.

Consider the system described above. A set of n nodes are changing the weights in the ranking matrix $\mathbf{W}$ iteratively using gradient descent with learning rate λ. $\mathbf{W}^{t+1} = \mathbf{W}^t + \lambda \Delta \mathbf{W}$. Here, the change of weights is $\Delta \mathbf{W} = [\Delta w_0, ..., \Delta w_n]$ where each Δw_i is a change to a single row pushed by node i. Each node is attempting to competitively maximize it's payoff as a function of the weights $P_i(\mathbf{W})$.

13.5.1 Definition

The ex-post regret for a single step is the maximum difference in loss between the chosen step Δw_i and all alternative Δw_i^*. The *expected* ex-post regret is this difference in expectation, where the expectation is taken over all choices Δw_j's chosen by other participants [16].

$$\text{rgt}_i = E_{\Delta w_j} [\max_{\Delta w_i^*} [P_i(\Delta w_i^*) - P_i(\Delta w_i)]] \tag{30}$$

13.5.2 Theorem

For sufficiently small λ, the expected ex-post regret for strategy $\Delta w_i = \frac{\partial P}{\partial w_i}$ is 0.

13.5.3 Proof

Consider Taylor's theorem at the point $\mathbf{W}$ for the payoff function P under a change in weights $\mathbf{W}^* = \mathbf{W} + \lambda \Delta \mathbf{W}$. There exists a function $h(\mathbf{W}^*)$ such that in the limit as, $\mathbf{W}^* \to \mathbf{W}$ we have the exact equivalence:

$$P(\mathbf{W}^*) = P(\mathbf{W}) + \frac{\partial P}{\partial \mathbf{W}} (\mathbf{W}^* - \mathbf{W}) + h(\mathbf{W}^*) \tag{31}$$

Let $P(\mathbf{W}_*)$ represent the payoff when the weight change of the i^{th} row is $\Delta \mathbf{W}_i = \frac{\partial P}{\partial \mathbf{W}_i}$, and let $P(\mathbf{W}^*)$ be any other choice. Since $\lambda \to 0$, we have $\mathbf{W}^* \to \mathbf{W}$ and by the definition of regret we can write:

$$\text{rgt}_i = E_{\Delta \mathbf{W}_j} [\max_{\Delta \mathbf{W}_i^*} [\frac{\partial P}{\partial \mathbf{W}} (\mathbf{W}^* - \mathbf{W}) - \frac{\partial P}{\partial \mathbf{W}} (\mathbf{W}_* - \mathbf{W})]] \tag{32}$$

This follows by subtracting Equation (31) with choice $\mathbf{W}^*$ and $\mathbf{W}_*$. Next, substituting $\mathbf{W}^* - \mathbf{W} = -\lambda \Delta \mathbf{W}$ and expanding $\frac{\partial P}{\partial \mathbf{W}} \Delta \mathbf{W} = [\frac{\partial P}{\partial \mathbf{W}_0} * \Delta \mathbf{W}_0, ... \frac{\partial P}{\partial \mathbf{W}_n} * \Delta \mathbf{W}_n]$ into the equation above leaves:

$$\begin{aligned} \frac{\partial P}{\partial \mathbf{W}} (\mathbf{W}^{\cdot} - \mathbf{W}) - \frac{\partial P}{\partial \mathbf{W}} (\mathbf{W}_{\cdot} - \mathbf{W}) = \lambda (\frac{\partial P}{\partial \mathbf{W}_i} \cdot \Delta \mathbf{W}_i^* - \frac{\partial P}{\partial \mathbf{W}_i} \cdot \Delta \mathbf{W}_{i*}) + \\ \lambda \sum_{j \neq i}^{n} (\frac{\partial P}{\partial \mathbf{W}_j} \cdot \Delta \mathbf{W}_j^* + \frac{\partial P}{\partial \mathbf{W}_j} \cdot \Delta \mathbf{W}_{j*}) \end{aligned} \tag{33}$$

The constant λ can be removed and the second term depends only on weights of other rows $\mathbf{W}_{j \neq i}$. Since these are independent and evenly distributed these can be removed under the expectation $E_{\Delta \mathbf{W}_j}$. He have:

$$\mathrm{rgt}_i = E_{\Delta \mathbf{W}_j}[\max_{\Delta \mathbf{W}_i^*}[\frac{\partial P}{\partial \mathbf{W}_i} \cdot \Delta \mathbf{W}_i^* - \frac{\partial P}{\partial \mathbf{W}_i} \cdot \Delta \mathbf{W}_{i*}]] \tag{34}$$

Finally, we use the the fact that for vectors a, b and angle between them θ, the magnitude of the dot product is $|a||b|cos\theta$. This is maximized when the vectors are parallel $\theta = 0$ and $cos(\theta) = 1$. In our case, we have the maximum when $\Delta \mathbf{W}_i = \kappa * \frac{\partial P}{\partial W_i}$ for some constant $\kappa > 0$. Thus $P(\Delta \mathbf{W}^*)$ is maximize when $\Delta \mathbf{W}_i^* = \kappa * \frac{\partial P}{\partial \mathbf{W}_i}$. Since $P(\Delta \mathbf{W}^*) = P(\Delta \mathbf{W}_*)$ in the maximum, this proves the point.