Metheus

Fair Accountable Transparent

Machine Intelligence Audit

Ling Xiao

12th July 2025

Abstract

Presently there are two mega trends ripping through the world:

1. Artificial intelligence (AI) and how it permeates every aspect of our lives.
2. Currency and supply-chain multipolarity as the post-war peace dividend no longer yields.

Metheus builds a set of market protocols that anneals these two separate trends into a *united front*. On the AI front we stand with the public, and push open-governance of artificial intelligence as a fundamental right. This is a call for supply chain reform around the principles of digital sovereignty. Metheus builds tools that allow anyone to audit AI models, thereby converting passive consumers into active decision makers. Specifically:

1. Just as material commodities have physical supply chains, AI models are products of digital supply chains. It spans data collection, data annotation, architecture exploration, model development, and finally refinement towards end-user integration.
2. Just as people are judged by IQ and EQ metrics, there are similar measures for AI models. IQ corresponds to the quality of models on impartial performance metrics, while EQ measures models along socially normative standards of fairness and equality.

Metheus constructs protocols so that the public may audit AI agents according to openly defined IQ and EQ standards, over the entire model development process.

1. In this autonomously financed supply chain, auditing pays for model development. So that as the supply chain is audited, it is also financed.
2. Concurrently, capable auditors are rewarded for discovering performant models and data. Thus the relationship between public auditors and the model developers is not an adversarial one. Instead they collude to produce socially moderated AI agents.
3. Critically, open auditing creates a public repository of communally maintained data and models. Thereby promoting new digital commons and standards in the presently fragmenting digital Westphalia.

Since Metheus decentralized auditing is equipped with financial incentives, we tap the flow of this auditing circuit to construct an *Entropy Digital Currency*. It is an analogue to the Petrodollar in the AI supply chain, and denominates the fundamental quanta of value along this circuit: information. Thus Metheus positions itself in the second trend. However whereas others see multi-currency mercantilism as nations trading in sovereign currencies, Metheus rezones the playing field as industries trading in programmable domain-specific currencies, exchanged along commonly defined technical rails and standards.

In conclusion, Metheus develops protocols that allow people to realize their fundamental right of governing artificial intelligence. We then leverage this AI auditing-circuit to bootstrap an industry-specific digital currency. Together they define a unified techno-social horizon upon which all may reside and prosper.

Contents

CHAPTER 1

Introduction

1.1 Overview

Recent decades have witnessed deep ruptures in how the modern economy operates. In the old model, intelligent people design mindless machines, and together they produce the goods and services. All of which are ultimately denominated in: the Petrodollar. The new age is just dawning, but it is becoming increasingly clear that in the future, machine intelligence will provide many of the services, and design many of the "goods." Presently, the fate of the Petrodollar is under fierce litigation. But the outcomes of the early innings suggest that its last act will be a violent one. This is a sea change that will torch industries, set entire communities adrift, and unmoor nations.

> There are decades where nothing happens; and there are weeks where decades happen.

This is disruption par excellence, a climate change of social relations. The way of Metheus is to *place as many people on the right side of this change as possible.* We address the problem wholistically by fielding protocols that mediate:

1. how machine intelligence is developed and its quality audited;
2. how data providers are motivated, audited, and compensated;
3. how algorithms are moderated for its social implications;
4. how the general public may partake and directly profit from this auditing process;
5. and most importantly: how this burgeoning economy is monetized.

Here "monetization" does not mean "making money." Rather, it quite literally means measuring the value created in this underlying economy by minting new currencies. Said numéraire are recorded on a single-purpose decentralized ledger, native to the Metheus ledger is the token-pair:

1. $\mathfrak{Fat}$: the Fair Accountable Transparent stablecoin;
2. $\mathfrak{miat}$: the Machine Intelligence Audit Token.

> $\mathfrak{Fat}$ is the world's first **entropy digital currency**.

Together they measure the productive potential of the Metheus economy, one that is built on quantitative and normative introspection of machine learning algorithms. Thus $\mathfrak{Fat}$ is the world's first **entropy digital currency**:

1. its underlying commodity of choice is not petroleum but raw information;
2. its refineries are graphic processing units (GPUs);
3. its refined oil is trained models.

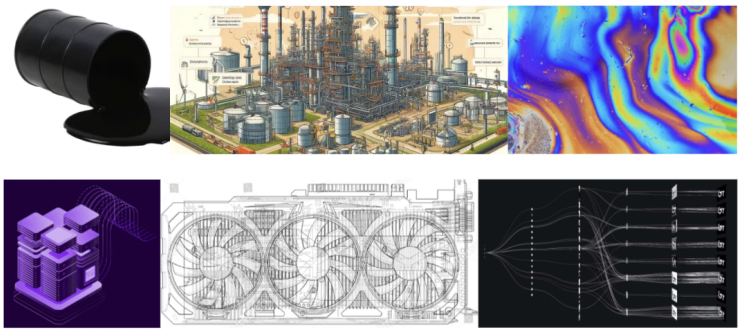


Figure 1.1: Metheus mints the fair, accountable, and transparent stablecoin ($\mathfrak{Fat}$). $\mathfrak{Fat}$ is the world's first entropy digital currency. Instead of feeding the perpetual violence of the Petrodollar economy, $\mathfrak{Fat}$ denominates the burgeoning machine intelligence world by auditing it.

This shift in underlying commodity of value is profound. Whereas the Petrodollar tracks population growth, the entropy digital currency harnesses growth in artificial intelligence. Whereas the Petrodollar is built upon perpetual conflicts, Metheus makes wagers, not war.[1] The moment to spring is now. This is clear in light of recent history.

Metheus makes wagers, not war.

1.2 Machine Intelligence

Much ink has been spilled over the transformative potential artificial intelligence (AI) ever since the first conversational agent ELIZA was conceived at MIT in 1964. Since then the field has seen its ups and downs. Highlights include:

1. from the disillusionment of neural networks in the 1980s due to insufficient data and compute power, to its 2014 redemption by internet-scale data lakes and GPUs.
2. From the utter infeasibility of solving GO with classical tree search, to its rousing resolution with deep reinforcement learning.
3. From the inconceivably futuristic sci-fi nature of human-like dialogue systems, to its now mundane applications due to massive generative models.

[1] Normally entropy means disorder. In this case entropy is a measure of information, see [Sha48].

Each trough of disillusionment is followed by the subsequent highs of a seemingly inevitable breakthrough. Once again genuine progress has been made in this cycle. Moreover, many of the AI models that enable this step-change are in the public domain, ready to be recapitalized on by intrepid opportunists (see remark 1.6.1). Presently, industrial scale artificial neural networks (ANNs) are generating images, videos, and holding conversations at a high enough fidelity to replace artists, human assistants, and even teachers. Consumer applications notwithstanding, the long-term trend of consequence is the massive investments at the infrastructure level. Since 2021, billions of GPU related hardware have been built by major corporations. However this is now followed by a growing sense of unease, as the current generations of generative models prove too error-prone for use-cases demanding high precision. [2] It is becoming increasingly clear that an entire crop of startups may not see revenue commensurate with the volume of investor money spent on training. As this consensus diffuses through the system, deal flow taper as investors conclude it is *they* who have been harvested by the likes of Microsoft. Meanwhile, Deepseek released the R1 model that is on parity with OpenAI's o1, at a mere fraction of the training and inference cost. This dramatic step-change fundamentally negates the thesis that massive compute capex is necessary precondition for winning the "AI race." SV companies scramble as they justify their massive fund raise and eye-popping valuation. The fear has spilled into main-stream media. The next AI winter must be coming.

However there are historical reasons to remain optimistic. This moment is similar to the burst of the dot-com bubble, whereby a torrent of broad-band infrastructure was built in anticipation of demand, which eventually came with the advent of web 2.0. In summary, the present juncture is as follows: there will be a deluge of compute capacity for model training and inference, and comparatively fewer customers using them. The pressing question for those who have sunk billions into GPU farms is how to survive this winter. They seek:

1. new use cases for existing models to recoup their astronomical training costs;
2. better ways of prospecting novel model architectures in anticipation of the next spring;
3. new data/model moderation tools to keep the regulators at bay.

Metheus offers one such avenue by bridging machine intelligence onto a decentralized environment. Thus tapping a broader capital base, new use cases, and new ways of socializing the technology.

Metheus bridges machine intelligence with DeFi capital, pegging the entropy-stablecoin to the open-source AI economy.

1.3 Decentralized Finance

Similar to artificial intelligence, decentralized finance (DeFi) as an industry has seen even greater boom and deeper busts.

1. In the first cycle, purist, libertarians, and anarchists showed that it is possible to build a permissionless payment system around Bitcoin.

[2] Overparameterized statistical models trained on past data tend to have high recall and low precision. This is fine in entertainment-related applications such as generating digital influencers, but may prove deadly in sensitive domains.

2. This is followed by a decentralized computational environment in Ethereum, featuring a native on chain currency Eth.

3. This new model of networked computation is complemented by a distributed storage environment in IPFS, featuring native currency filecoin.

However wild fluctuation in prices of Bitcoin and Eth render their value as a numéraire dubious. This has inspired a family of stablecoins whose value is pegged to 1 unit of any currency. Broadly speaking, there are many ways to peg a stablecoin:

1. (*Fully collateralized fiat backed stablecoin*): the users are essentially lending money to the platform by depositing 1 fiat of choice, and receiving one minted stablecoin in return. Tether is the most well-known example with a market cap of USD 138.3B. Although Tether can be redeemed in theory, in practice it is not straightforward. Moreover, since customer deposits are reinvested in difficult-to-audit ways, de-pegging is a very real risk [Har+20],[AS22].

2. (*Central Bank Digital Currencies or* CBDC): one subset of projects adjacent to the category above are CBDCs [D25]. As of 2023, over 100 countries representing more than 95% of global gross domestic product, are actively engaging in research, development, pilots or have fully-launched CBDC initiatives [SM+23]. The leader here is project mBridge, a permissioned blockchain project for whole-sale CBDC. Its primary purpose is intra-central-bank settlements, cross-border atomic settlement of tokenised primary issuance, and supply chain financing [[hkm22],[UBS21]].

3. (*Over-collateralized stablecoin backed by a "diversified" basket of cryptocurrencies*): similar to an exchange traded fund, the "vault controller" assembles a basket of cryptocurrencies and issues shares of the underlying tracking any fiat of choice. Stability here requires active market making. A more fundamental flaw is that DeFi coins are highly correlated, so this "diversified" basket may not even exist [AM+23].

4. (*Algorithmic stablecoins*): or Ponzi scheme. Luna is the most notorious example. It imploded after a catastrophic deleveraging spiral, and its founder is now in jail [KP+23].

Despite its arduous history, the utility of stablecoins stands true. Technically speaking, any fiat currency bridging onto a DeFi environment requires an on-chain mirror anyways. This unit of account is by construction a stablecoin. Furthermore, any usable decentralized economy must stabilize the price of its native currency. So Metheus pushes the field forward by pegging the $\mathfrak{Fat}$ stablecoin to a true economy with sustainable future value.

1.4 The Path Forward

Charting a path forward requires a keen understanding of the intrinsic rhythm of how both industries have progressed. In AI, there is an exploration-exploitation trade off in prospecting models. In the exploration phase, many architectures across data verticals are tried, so one must justify scaling the model to an industrial scale. [3] Once a viable model architecture is (re)discovered, the industry enters the exploitation phase. Here models are packaged and sold as enterprise software, and sometimes consumer products. While AI summer is exploitation, the subsequent winter is a time for exploration. Naturally, winter lasts longer than summer. So given its longer duration and requisite capital outlays, exploration is done in universities and well-funded industry labs.

Metheus reforms this process by supplementing open-source data and models with distributed governance and *open-source capital*. The Metheus ecosystem transforms industrial-scale data annotation and model exploration into a profitable decentralized protocol.

[3] For example, training dialogue systems using recurrent neural networks is not a new idea, but training them at the scale of OpenAI was.

Concurrently, we will address the structural issues around open machine intelligence governance that is only possible in a truly decentralized environment. So that the fruits of everyone's labor shall exist outside of major companies, in a way that the products of OpenAI and DeepMind did not.

AI as an industry is built on open-source models, free data, and transparent benchmarks. Metheus pushes it further with distributed governance, and open-source capital.

In DeFi the cycles are difficult to predict, so no attempt is made here. However, history suggests activities in this space are downstream to global events and local disruptions. It is quite clear that the next five years will bring a cluster of unexpected events across the globe that may even exceed the previous five years, this bodes while for the industry as a whole. Meanwhile, Bitcoin and Eth have gained institutional acceptance, so au contraire the detractors they are here to stay. Although $\mathfrak{Fat}$ is designed to be fiat-collateralized, the existence of a decentralized currency network and a persistent permissionless compute environment gives us additional legs to stand on. The groundwork is laid for a decentralized capital and compute environment for auditing machine intelligence.

1.5 Decentralized Machine Intelligence Audit

The blazing speed with which AI has developed in the last few years may rouse the technically inclined, but its "life-like" outputs has spooked main street and regulators alike. Rather than "move fast and down with the luddites," a wiser approach may entail bringing a broader swath of people into the research and development process. This is in accordance with the logic of history, where technology tend to be socialized in two waves:

1. first an aggressive first mover pushes the technology into mainstream consciousness as a viable mass-consumer product.

2. However, this first wave is often deemed too aggressive or even illegal. So what follows is the slower next wave that benefits more parties, thus finding long-term success.

The path from Napster to Spotify is the canonical example. Metheus follows this wave of generative AI by socializing passive users into active decision makers. They negotiate the nature of this technology by way of *auditing*. Now the word "audit" has a precise definition in the field, however we pursue a more expansive interpretation, this allows Metheus to protocol a larger segment of the AI development lifecycle.

Definition 1.5.1. (*Machine Intelligence Audit*): is the quantitative and qualitative introspection of machine learning models, the data they are trained on, and the entire development process. This process proceeds along all conceivable dimensions, such as:

1.5.1.1. *full spectrum quality audit*: of data to be used to train models.

1.5.1.2. *performance introspection*: along well-defined metrics of models trained on data.

1.5.1.3. *annotator assessment*: of people who provide labels to data.

1.5.1.4. *fairness, equality, and equity inspection*: according to fluid and socially defined norms of what is "acceptable" behavior for AI agents.

The vectors defined here are far from exhaustive. Moreover, since the community is constantly developing new and better metrics to characterize each point, the variety of auditing protocols is in effect unbounded. In more basic terms, "auditing" is fundamentally a *protection mechanism*. Metheus steers the relationship the general public has with machine intelligence, and protects the public from the prospect of intelligent machines. All the while giving the people a chance to profit from this arrangement.

Decentralized auditing is a public safety mechanism. Metheus mediates people's relationship with machine intelligence, and protects the public from the prospect of intelligent machines.

In allegorical terms, "auditing" is AI governance. Whereby the machine learning algorithms are the governed, while the people form the government. From a positioning standpoint, auditing as a basis for open protocol is well situated amongst industry stakeholders:

1. *Model foundries*: we complement well-capitalized foundational model developers by *social proofing* AI in a public certification process. People quite literally buy in.
2. *Compute vendors*: auditing complements their hardware and improve their revenue outlook. The Metheus auditing process creates communally maintained pools of data and models that are continuously updated. At the transactional level, auditing is a prolonged inference session, so it smooths out the demand curve for existing AI models.
3. *The public*: auditing gives the average person a stake in the algorithm development process, it is a new way of relating to AI that may become mainstay of culture. Think nutrition labels on the back of food packages that have become legislated must-haves.
4. *Regulators*: their attempts to regulated AI as new abstraction over the internet is understandable, and their contempt towards DeFi is a perennial given. Metheus gives regulators a way to steer one tech cycle with tools from another, in effect "using poison to fight poison."

Fair and accountable AI is the new ESG

1.6 Metheus System Overview

System Components

This section outlines the silhouette of the Metheus economy, with a particular emphasis on its main components.

Data Centers and GPU Farms

Infrastructure-level support will enter the Metheus ecosystem through third-party vendor partnerships. They are used to host Metheus-associated models and data. They *will not* be used to mine cryptocurrency.

Metheus Consortium (Side) Chain

On the technical front, the consortium sidechain is a refactored version of Ethereum, whereby the native Eth is removed to prevent speculation. It is the settlement layer for all market making activities in Metheus. Critically, it also hosts the primary-market $\mathfrak{Fat}$ stablecoin. Price action on the Metheus sidechain keeps the $\mathfrak{Fat}$ price pegged to 1 fiat of choice. On the partnership front, the Metheus consortium chain is the technical fabric that binds a new form of Blockchain Industry Association: one that is built around the notion of AI governance. Vendors along the entire AI supply chain can participate following this template:

> "consortium chains would typically involve those active in, and responsible for, the production, distribution, storage and transport of commodities in question, as well as other stakeholders such as regulators and financial services actors." [Pow23, p. 132]

In the context of AI industry, the commodity in question are data and models, as well as resellers and distributors of any hardware components. The consortium members determine how the chain should operate, observing the tradition whereby:

> "membership of associations can readily map to a technical topology, whereby different members with varying authorities can perform different kinds of functions in the overall information system." [Pow23, p. 136]

There is an analogue of this very dynamic in the Riskstream collaborative, which proxies the entire insurance industry. It defines the compliance needs of industry players, trains the insurance agents, certifies them, and fields a consortium chain for settling claims. The difference is that where as Riskstream is a cost center to which insurance companies must pay their dues, Metheus is a revenue generator that finances the development of AI models as they are audited.

Mainnet

Primary-market $\mathfrak{Fat}$ tokens on the Metheus sidechain factors into two secondary-market tokens on a production mainnet environment. Any EVM compatible chain will suffice. This process is depicted in figure 1.2, and is a critical component of the incentive package geared towards Auditors. $\mathfrak{Fat}$-α in particular captures the upside of the underlying AI audit economy mediated by the Metheus market protocols. See chapter 4 for more information on how $\mathfrak{Fat}$-α and $\mathfrak{Fat}$-β are valued.

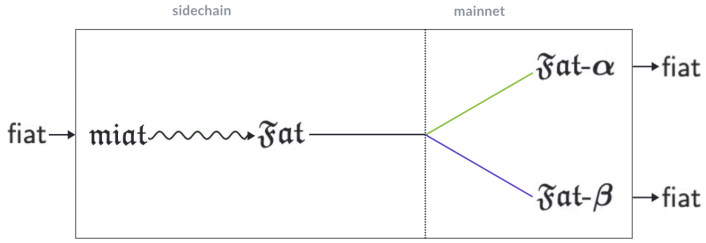


Figure 1.2: The sidechain stablecoin $\mathfrak{Fat}$ captures the economic upside of the underlying Metheus economy due to fluctuations in miat supply. The upside of the underlying is passed onto $\mathfrak{Fat}$-α, while $\mathfrak{Fat}$-β remains the fiat-pegged fully collateralized stablecoin.

Metheus Market Protocols

Figure 1.3: In traditional machine development pipelines:

1. data is collected and sanitized by private firms as "back-office" work;
2. annotation is sometimes outsourced completely to third parties;
3. since data carries biases that percolate into the models trained on them, the data itself is also audited along social-compliant measures;
4. foundational model development is done by private firms, and are published only after they have achieved some level of performance.

Metheus opens up this lifecycle completely, so that the "kitchen is in the dining room," so to speak. Metheus market participants now:

1. audit the data w.r.t to fairness/quality measures, and profit in doing so.
2. Model development occurs in the open in competition-like environments. Model "checkpoints," are tokenized into mints, and marketed to Auditors to fund model development.
3. Auditors trade mint tokens to judge their relative performance against other models in the same class.
4. Model output are also audited for fairness and equity infractions. The users' judgments are tokenized and marketed for compensation.

Metheus fields two major classes of auditing protocols:

1. QAudit: or machine intelligence quality audit. This protocol satisfies the "full spectrum quality audit" and "performance introspection" of definition 1.5.1. In particular, QAudit mediates both:
   a) *data markets*: where participants can sell their data to third parties, so that the quality of said data is evaluated as they are listed.
   b) *model markets*: where a distributed team of machine learning engineers coalesce to develop AI models. This setting is similar to Kaggle, however Metheus incentives are significantly better for all stakeholders involved.

   From a user-interface standpoint, QAudit as a product is a new kind of computational environment where AI engineers do not pay to train models. Rather they are *paid* to train models when they push intermediate states of the unfinished model, or "*checkpoints.*" The more checkpoints pushed, they more they they are paid. [4]

[4]The Metheus QAudit protocol is based on work done at Harvard, funded by the Ethereum Foundation [[HW19], [WFA15]]. Amazon AI Labs is also fielding a variation of ensemble crowd-sourced machine learning competition [GH+24].

2. FeaAudit: satisfies the "annotator assessment" and "fairness, equality, and equity inspection" of definition 1.5.1. It socializes the broader populace into AI development by giving them a consumer-facing annotation protocol. However FeaAudit is far from some trivial pursuit, it is the lever to shape the "social intelligence" of AI software. The nature of FeaAudit depends on the perspective of the participants:

   a) from the perspective of annotators, FeaAudit is a game whereby they answer questions to survive an elimination game, and are paid if they "hit a run" of acceptable answers.

   b) From the perspective of the market owner, FeaAudit is an annotation service whereby they are not paying for data labels, but rather are *paid* while procuring data labels.

In simpler terms, QAudit is model and data development, while FeaAudit is data annotation and annotator quality assessment. Together, they protocol a significant portion of the data, model, and product development process (see fig 1.3). Any Metheus participants can launch *instances* of these protocols in a permissionless manner, so as to capitalize their own AI development pipeline. Hence the name "decentralized audit." Finally, auditing is a persistent process that never terminates. "Cooking" the product in the open demystifies machine learning, and the public will see that AI agents are fundamentally huge regression models with limited operational domain. [5]

While others profit from the public's ignorance, **Metheus** benefits from the public's vigilance.

Metheus Token Ensemble

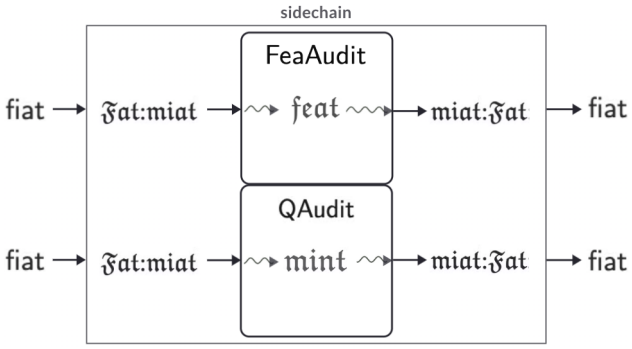


Figure 1.4: In the Metheus economy, fiat is converted to $\mathfrak{Fat}$ on 1:1 basis, and then converted to ERC20 token $\mathfrak{miat}$ at a floating exchange rate with partial bonding curve (see fig 4.7). This $\mathfrak{miat}$ buys $\mathfrak{mint}$ in QAudit markets, and $\mathfrak{feat}$ in FeaAudit markets. Both are intermediate currencies issued and consumed within one audit session, where their $\mathfrak{miat}$-denominated upside is realized in *that* session. This $\mathfrak{miat}$ exits the sidechain at spot $\mathfrak{miat}$:$\mathfrak{Fat}$ exchange rate, and convert back to fiat at 1:1. See fig 1.5 for full conversion lifecycle.

[5] One aspect of AI development we do not touch is the productization thereof, especially when it involves with company's private data.

The underlying commodities in QAudit and FeaAudit markets are denominated with:

1. (mint): *the machine intelligence nonfungible token*, akin to an ERC-1155. This is effectively a share of model or data. It denotes the quality of the underlying commodity. An overview of its tokenization process is provided in remark 1.6.1. mint tokens are denominated only in miat.

2. (feat): *the fair and equal annotation token*, akin to an ERC-721. This denotes the quality of an annotator or Witness. It is also denominated only in miat.

Referencing figure 1.4, we see that:

1. if the underlying commodity performs, then mint and feat can be redeemed for fiat at well-defined market prices. Thus they have *exchange value*.

2. Because feat and mint can only be purchased with miat, they also endow miat with *exchange value*.

In sum, miat, mint, and feat are *currencies*. Meanwhile, the underlying data, labels, and models possess clear use value. Additionally, unlike nonfungible tokens of the last cycle, mint and feat do not have infinite lifespans. Instead they are:

1. issued at specific moments within QAudit/FeaAudit market lifecycles;
2. have *bounded* upside that is rigorously defined per protocol rules;
3. and most importantly: have fixed expiration dates.

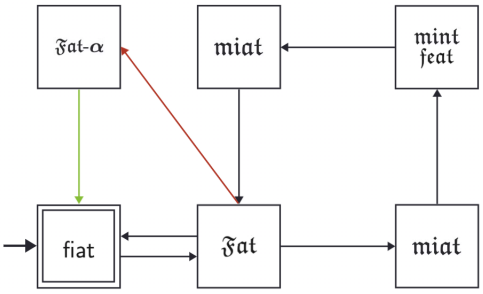


Figure 1.5: The conversion cycle begins when Auditors enter the market with fiat, and proceeds along the path: fiat → Fat → miat → (mint, feat) → miat → Fat → fiat. Observe the red arrow from Fat to Fat-α, this is known as miat-*inflation-dividend*. The upside of which accrues to holders of Fat-α tokens. See chapters 2 and 4 for full discourse.

Figure 1.5 depicts the token ensemble conversion lifecycle in the Metheus economy, observe:

1. the supply of Fat and miat are unbounded, however their rate of growth is not.

2. In particular, the growth in miat supply is a function model, data, and annotation volume in the Metheus ecosystems. It is a measure of the system's capacity to process information. Its rate of growth can never exceed this capacity.

3. Concurrently growth in Fat supply is proportionate to fluctuation in miat supply, and its induced miat-inflation dividend. Just as miat supply measures "temperature" of the underlying AI economy, Fat is the "entropy digital currency" of this economic system.

4. Moreover, relative Fat and miat supply fluctuates within a fixed exchange-rate envelope (see fig 4.7). This envelope maintains the positive miat-inflation invariant.

5. Critically, the ceiling and floor of this exchange rate envelope are determined by $\mathfrak{Fat}$ and/or $\mathfrak{Fat}$-$\boldsymbol{\beta}$ holders. So $\mathfrak{Fat}$-$\boldsymbol{\beta}$ is not just a fiat-collateralized stablecoin, it is also a *governance token* that determines the allowed inflation dividend $\mathfrak{Fat}$-$\boldsymbol{\alpha}$ holders receive.

In sum $\mathfrak{Fat}$ and $\mathfrak{miat}$ are *loosely coupled*, their fluctuating exchange rate contribute to the $\mathfrak{miat}$-inflation dividend of fig 1.5. This is the source of $\mathfrak{Fat}$ stablecoin's upside. This monograph will show that Metheus not only improves DeFi tokenomics of past projects, it also betters the incentives of AI-development as well. See chapter 2, 3, and 4 for in-depth discussion.

Market Participants

The core Metheus protocol addresses a spectrum of stakeholders, with different profiles from retail inclined to professionally oriented. The professional entrants are:

1. (PA principle auditor): they launch instances of QAudit and FeaAudit with the intention of procuring/inspecting data sets or models.

2. (Agents): these can be teams of machine learning engineers. They develop models in context of QAudit-machine learning markets, and receive fiat-denominated subsidies on training, in exchange for pushing their model checkpoints into the QAudit competition.

The retail side include:

1. (Auditors): these are retail DeFi players who seek new tokens to trade. The principle contribution of Metheus protocol is massaging their trading activity into socially-positive auditing activity.

2. (Witnesses): they are FeaAudit annotators who judge AI agents for fairness infractions. Their roles can be further generalized to annotation tasks of any kind. The FeaAudit-Witness paradigm is a direct substitution for Amazon mechanical Turk farm. However unlike other annotation services, in FeaAudit the annotators are not paid, instead they "pay to label." Chapter 3 shows how this is possible.

3. (Agents): in addition to being teams, they can also be consumers who have data to sell or annotate. In which case they can deposit their goods into a QAudit-data-market instance for Auditor-judgment, and receive fiat rewards.

Finally, there is the $\mathfrak{Fat}$-creditor. This monograph covers $\mathfrak{Fat}$-offerings suitable for professional and retail counterparties. Ideally most $\mathfrak{Fat}$ holders are retail, thereby socializing the upside of the AI economy broadly. This is not an euphemism: $\mathfrak{Fat}$-creditors should never spend more than 1 fiat of choice per $\mathfrak{Fat}$ stablecoin, and its value is always *at least* 1 fiat. Chapter 4 provide a rigorous valuation framework for $\mathfrak{Fat}$ stablecoins.

Remark 1.6.1. (QAudit *bespoke tokenization process*). Metheus is a protocol developer that enable principle auditors (PA) to launch their own market instances. [6] However these markets can be complex. QAudit for example is a multi-sided market, cold-starting them is challenging. The most difficult set of participants is the Agents, they must agree to submit models for wager on the public market. However Agents are professional algorithm developers with well-defined workflows, they may not even be allowed to engage in "DeFi activities."

This is why QAudit can run in "single-set-of-player" mode, where the three sided market of PA, Agents, and Auditors is reduced to PA and Auditors. The PA now just has to convince

[6] In crude terms, the PA owns the market instance; i.e. Jeff Bezos own Amazon, so he would be a PA. The Amazon sellers would be the Agents, while the buyers are the Auditors. QAudit is set up so that Amazon does not need sellers, just Bezos and buyers.

Auditors (read: DeFi players) to wager on existing open-source models. Figure 1.6.1 below describes how QAudit can be configured for this task.

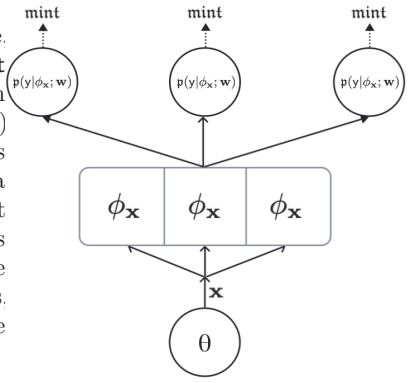


Figure 1.6: QAudit "single-set-of-player" mode, read from bottom to top. 1) The QAudit mint tokenization process samples data $\mathbf{x}$ from generative model $\boldsymbol{\theta}$, with automated label y. 2) This data embeds into 3 different feature spaces denoted in $\phi_{\mathbf{x}}$. 3) Each set of featurized data trains 3 classifiers $\mathfrak{p}(y|\phi_{\mathbf{x}}; \mathbf{w})$ with different objectives. 4) QAudit tokenizes the classifiers $\mathbf{w}$, not the generative model $\boldsymbol{\theta}$. Critically the generative model $\boldsymbol{\theta}$ need not come from Agents, as there are many open-source models available for tokenization. See [Hug24] for examples.

In the example of figure 1.6.1, given one class of open source generative model, the PA launches $3 \times 3 = 9$ different *derivative* QAudit markets wagering on their quality. The protocol trains the classifier $\mathbf{w}$'s, so no Agents participation is needed. This is not just business sense, there are strong technical reasons for this tokenization process. See chapter 2, Appendix D, E, and F for full technical discussion.

Operating Model

This section ties the various factors together into the Metheus operating model. Referencing fig 1.7, reading from top to bottom. The model proceeds as follows:

1. ($\mathfrak{miat}$ mine): in the Metheus machine intelligence auditing economy, $\mathfrak{miat}$ is mined from a global pool at a rate commensurate with market activity platform wide.

2. (Audit markets): when Auditors wager in QAudit and FeaAudit market instances, they purchase $\mathfrak{miat}$ with fiat of choice at the current fiat:$\mathfrak{Fat}$:$\mathfrak{miat}$ conversion rate. Then they use $\mathfrak{miat}$ to buy model shares $\mathfrak{mint}$, and Witness shares $\mathfrak{feat}$.

3. (Inflation pool): as shares are traded and $\mathfrak{miat}$ mined, the markets realize both fees and $\mathfrak{miat}$-*inflation-dividend*. The fees are taken by the Metheus operating company, while the $\mathfrak{miat}$-inflation-dividend are placed into the inflation pool. Critically, since the Auditors buy into the QAudit/FeaAudit market with fiat, this inflation dividend is denominated in fiat, not $\mathfrak{miat}$. The inflation pool is an off-chain traditional finance (TradFi) facility.

4. (Collateral pool): Metheus then issues $\mathfrak{Fat}$ stablecoin, pegged to fiat of choice. This $\mathfrak{Fat}$ is partially collateralized by the Metheus operating company, with additional contributions from $\mathfrak{Fat}$ creditors. Note this means the $\mathfrak{Fat}$ creditor is spending less than 1 fiat to purchase 1 $\mathfrak{Fat}$ stablecoin of that denomination. However given Metheus's contribution, $\mathfrak{Fat}$ is still fully collateralized. This bears repetition: Metheus will not debase the $\mathfrak{Fat}$ stablecoin. The fiat collateral are placed in off-chain TradFi credit facility, where KYC/AML is part of the $\mathfrak{Fat}$ issuance process.

5. (TradFi bridge): the collateral pool places deposits in safe assets, and earn yields. The yields fund Metheus, while $\mathfrak{miat}$-inflation-dividends are given to $\mathfrak{Fat}$ or $\mathfrak{Fat}$-$\boldsymbol{\alpha}$ holders as seen in step 3.

In summary, $\mathfrak{Fat}$ holders trade yield in their fiat currency of choice for $\mathfrak{miat}$-inflation-dividends. They are swapping one low-yield but stable source of interest, for a high-yield high-risk instrument in $\mathfrak{Fat}$-$\boldsymbol{\alpha}$. However the value of their $\mathfrak{Fat}$-$\boldsymbol{\beta}$ never depreciates modulo TVM.

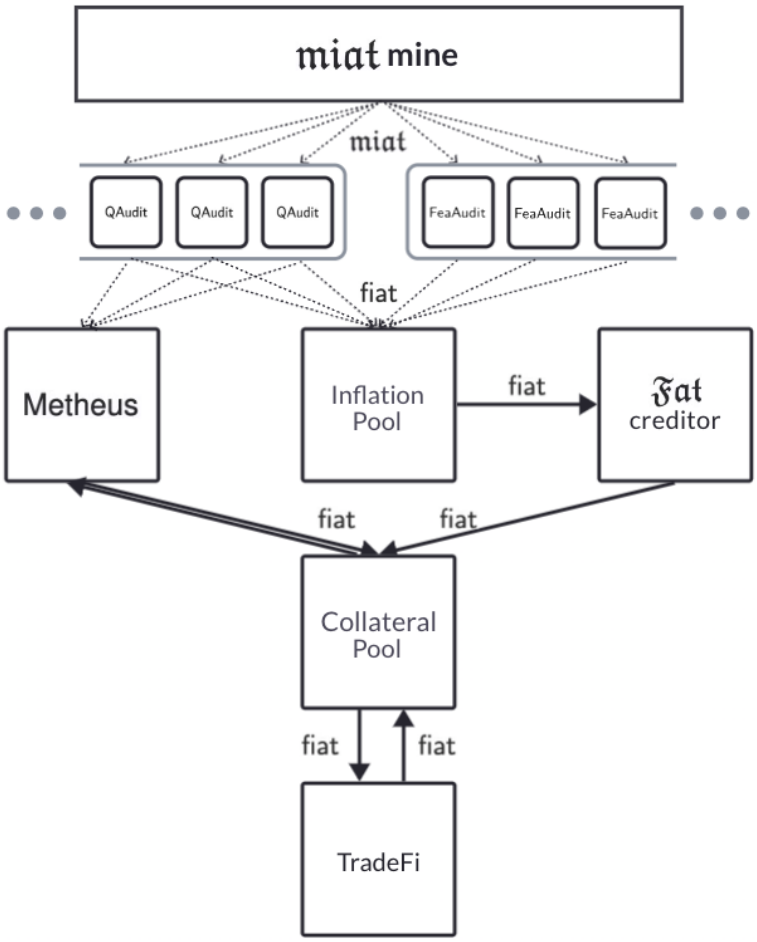


Figure 1.7: The Metheus operating model.

System Calendar

The operating diagram of fig 1.7 can be further enriched by a model of the Metheus's operating calendar. In this view, Metheus is an orchestrator that syncs the rhythm of the DeFi sector with the development process of AI models. This occurs most in QAudit, which proceeds along well-defined *stages*, whereby in each stage models are submitted by Agent, wagered on by Auditors, and the preliminary winners announced. $\mathfrak{Fat}$-$\boldsymbol{\alpha}$ distribution tracks this submission/wager cycle, so that $\mathfrak{Fat}$-$\boldsymbol{\alpha}$-holders receive regular payouts.

Moreover, QAudit $\mathfrak{mint}$ auction rules are such that $\mathfrak{mint}$ token prices oscillate along predictable trajectories over this calendar, in an overall sweep that resembles Bitcoin. The Metheus market protocol allows a *controlled* amount of speculation, but resets the market on a regular schedule so that prices do not exceed the use-value of the underlying. This is depicted in the top curve in fig 1.8. Predictable price oscillation is part of the broader Metheus productization process, whereby commodities such as data, models, and annotations

from one sector of the economy are presented in an attractive manner, so as to appeal to market participants from the DeFi world.

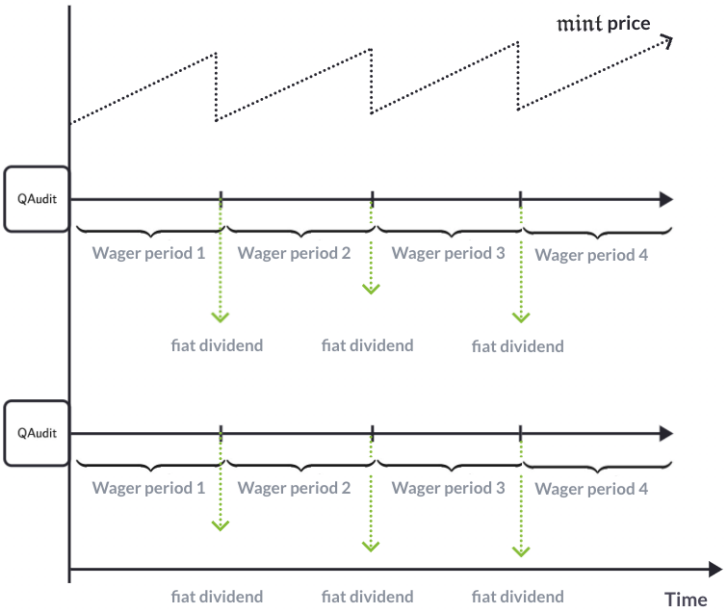


Figure 1.8: The top level user interface of Metheus is the system calendar, whereby multiple QAudit competitions and FeaAudit oracle sessions are running (only QAudit instances shown). Each one features wagering session that may last 1 week, where Agents submits model checkpoints or raw data, and Auditors wager on the quality of models/data submitted. Each weekend, their positions are liquidated. The winning proceeds are distributed, and $\mathfrak{miat}$-inflation dividend paid out to $\mathfrak{Fat}/\mathfrak{Fat}$-$\alpha$ holders. This AI-auditing calendar syncs the rhythm of DeFi players with AI model development cycles, and ensure the activity of one sector (DeFi) subsides the expenditure of another (AI). In Metheus, *trading subsidizes training*.

Philosophically, one may conceptualize QAudit as standardized exams for AI models, while FeaAudit instances are standardized quizzes for annotators. Then the "Metheus Calendar" is in fact a regular examination schedule. This is a very old institutional design pattern that collates talent from afar and near: be it performant AI models, machine learning engineers, or quality annotators. In the very long run, the Metheus decentralized institution may contend with deep-pocketed Industry Labs for the future of AI development. This includes new talent, new technical standards, and new culture of machine learning research and production.

System Incentives

We have characterized two ways of viewing Metheus: 1) as a set of markets protocoled by QAudit and FeaAudit, and 2) a calendar. The third way is to view Metheus as a system of incentives - this is the product in its purest form. In QAudit you will find these principles:

1. Trading subsidizes training.
2. Agents: eat what you bartered with.
3. Auditors: eat what you paid for. But if you defect, you shall regret.
4. Auditors and Agents hunt together, but eat separately.

In FeaAudit we have desiderata:

1. The principle Auditor is hidden.
2. The vice Auditor is exposed.
3. The Witness is hidden.
4. The Cohort is exposed.

These are no mere normative statements, rather they are defined precisely along mathematical lines. In the main body and Appendix C, we show these principles prompt *incentive compatible* behavior that yields a socially positive **decentralized market consensus mechanism**, and a **privacy-preserving data-release mechanism**. And finally in $\mathfrak{Fat}$ issuance we simply incite an arrangement so that **auditors are creditors**. This last tenet is critical, it ties the system end-to-end so that the most challenging stakeholders are incentivized to hold $\mathfrak{Fat}$ for the long run, thus improving the quality of activity system wide.

1.7 How to Read this Monograph

This ends the overview to the Metheus ecosystem. The rest of the monograph will see a marked tonal shift. You will find a rigorous layer-by-layer deconstruction of Metheus protocols in QAudit, FeaAudit, and $\mathfrak{Fat}$-issuance mechanism. The audience who trolls arxiv.org and/or are used to reading technical documents will find it a breeze. Those who are at home in commercial whitepapers will find this book dense. Nevertheless, we hope this introduction piqued your interest sufficiently, so that both sets of audience give it an honest attempt. If you are reading this monograph for the first time, you may find this order amenable:

1. Chapter 2 on QAudit: read section 2.1 to understand the basic rules of the game. Then pay particular attention to the tables in section 2.2 for how price and rewards are set in the QAudit staggered sequential parimutuel prediction market. You may skip the text and code block in sections 2.3 and 2.4, however skimming the diagrams and their caption will enrich your understanding of protocol details. Pay close attention to examples 2.3.1 and 2.3.2 to understand Agent and Auditor payouts under $\mathfrak{miat}$:$\mathfrak{Fat}$ conversion. The proof sections can be skipped entirely. However you should read definition 2.6.8 on *regime change*, which is an important market mechanism. Finally read section 2.7 as it is short, and it describes how auditing could be stretched into a perpetual process that 1) does not terminate, and 2) precipitously elevate $\mathfrak{mint}$ token prices per market participation.

2. Chapter 3 on FeaAudit: read section 3.2 to understand why FeaAudit is a necessary process in the context of Metheus ecosystem. Read the diagram captions to understand the FeaAudit annotation game. Section 3.4 describes how the FeaAudit oracle can be tokenized. You may skip the math equations if you are not used to reading dense notation, however the text blocks are approachable and should be read.

3. Chapter 4 on $\mathfrak{Fat}$ issuance: if you are reading this monograph to understand the $\mathfrak{Fat}$ stablecoin, then this chapter may be read in full. In particular, pay close attention to section 4.3 to understand how the conversion bonding curve affects $\mathfrak{fiat}$:$\mathfrak{Fat}$:$\mathfrak{miat}$ exchange rates. This monograph also offers a valuation framework for $\mathfrak{Fat}$-$\boldsymbol{\alpha}$ in section 4.4. This valuation method is in fact the Metheus *system dynamics*, so it reinforces the high-level ecosystem view we presented here, albeit in more precise terms.

If you are technically inclined and wish to examine this monograph closely, then there is an alternative reading order for you. In the inverted order, this monograph specs out an academic discipline or curriculum. This discipline ingests a machine-learning based economy, and outputs a stablecoin pegged to any fiat currency of choice. The order is as follows:

1. Appendix F: derives the exponential family distributions. The convex property of this family's partition function A is the basis for many proofs in this monograph. This chapter should be supplemented by books [WJ08] and [WJ04].

2. Appendix E: is an introduction to machine learning on a *narrow* pedagogical track. In this path, we dispense with the normal curriculum and focus on kernel-based methods only. In particular, we review recent work on *neural tangent kernels* in E.4. The knowledge here is not necessary to understand QAudit per-se, but it does help you understand the basic motivation for its price, reward, and tokenization process. This chapter is supplemented by [GB16], [JM24], [SC04], [SS16], and [RW06].

3. Appendix D gives a minimal introduction to the theory of sequential prediction markets, especially as it applies to machine learning algorithms. It motivates our tokenization process whereby the discriminative proxy is judged and tokenized in `mint`. This chapter should be supplemented by [CV10], [AK+14], [Kut15], and [ACV12].

4. Appendix B gives some of the basic definitions used in the proofs. They are drawn from [MR95], [Tad13], and [Rou14]. Observe many of the definitions on games are lifted from computer science, not microeconomics.

5. Now read chapter 2, 3, and 4 in context of the proofs in Appendix C, for a deeper understanding of this discipline. Although we did not conceptualize FeaAudit along orthodox lines as it is a consumer product, it should be read in the context of the canonical treatments found in [MP+24] and [DR14].

Finally in Appendix A you will find the major notations used in this monograph.

Metheus:
Open Source AI
Open Source Capital
Open AI Governance

PART I

Core Protocol

CHAPTER 2

Machine Intelligence Quality Audit

2.1 Introduction

Metheus does not market one single protocol, but rather deploys a *family* of protocols, each introspecting machine learning (ML) systems through a different lens. The foundational protocol in Metheus evaluates the performance of ML algorithms under certain pre-defined performance metrics. This protocol runs a set of machine learning competition markets akin to Kaggle, albeit under significantly better incentive structure. Unlike Kaggle which can be winner take all, every Metheus market participant who submits model or data "in good faith" receives awards commensurate with their contribution. The competition proceeds as follows:

1. Any *principle auditor* or PA may initiate an auditing market by launching an instance of the *machine intelligence quality audit protocol*, or QAudit. The PA launches the auditing process with the intention of procuring models or recruiting top machine learning talent, using the model performance as judge. The QAudit protocol is agnostic to the underlying submission so long as it emits data in the right format. Thus the PA may also launch a QAudit instance sourcing raw data, in which case this particular QAudit session is a *data market*.

2. The PA deposits:

   a) a fiat-denominated bounty to the protocol;
   b) defines how many *stages* T the QAudit session lasts;
   c) the time duration of each stage t.
   d) how many winners R there are at *every stage* t of the competition.

   For example, if an audit session may span 1) $\mathsf{T} = 60$ stages, 2) each stage is 24 hours, 3) it admits $\mathsf{R} = 3$ top teams out of a total of 100 teams, and 4) the bounty is $\mathfrak{v}_{\mathsf{pa}}$, then we denote this audit session $\mathsf{QAudit}(60, 24, 100, 3, \mathfrak{v}_{\mathsf{pa}})$. Submissions are judged in a preliminary manner as they are submitted in every stage t. But rewards are given after the competition ends, when all the models are evaluated against a PA-provided test set.

3. The Agents or *teams* then submit their model into the competition. Then in every stage t, each Agent's model is judged against all other Agents' model submitted within *the same stage*. In the $\mathsf{QAudit}(60, 24, 100, 3, \mathfrak{v}_{\mathsf{pa}})$ instance, QAudit admits daily winners announced *ex post facto* at market close after 60 days. Note that $\mathsf{R} = 3$ implies 3 Agents win every stage for 60 stages, so that there are $3 \times 60 = 180$ winning Agents per $\mathsf{QAudit}(60, 24, 100, 3, \mathfrak{v}_{\mathsf{pa}})$ session, along with the Auditors who wagered on the winning Agents' models. Moreover, neither the losing Auditors nor the losing Agents in each stage are eliminated. So an Agent that lost one day can always win the next day. Agents can win as often as they perform well, so that i.e. it is possible for an Agent to win 60 times. See figure 2.10 for a visual representation of how models are compared.

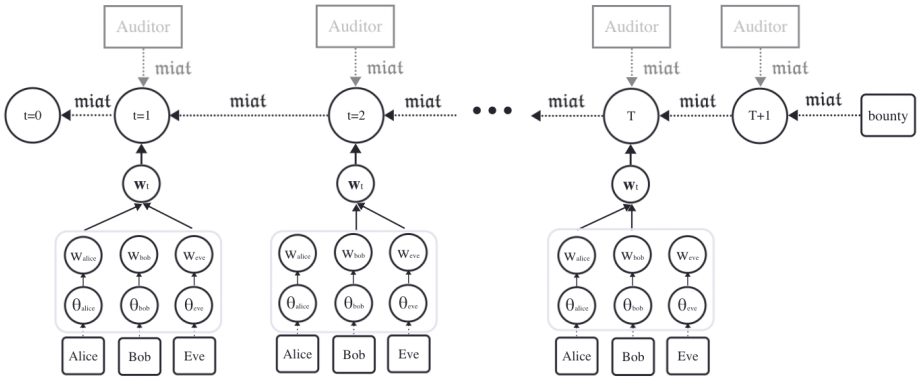


Figure 2.1: QAudit is a staggered sequential parimutuel market where in every stage t, Agents such as alice, bob, and eve submit generative models $\boldsymbol{\theta}$, while Auditors wager on these models with miat. The data sampled from each generative model is used to train a corresponding discriminative model $\mathbf{w}$ as part of the tokenization process. The best performing discriminative models in every stage is summarized by an ensemble discriminative model $\mathbf{w}^{\mathsf{t}}$. The winning Agents in each stage t are paid by those Auditors who wagered on the losing Agents on the same stage t. The winning Auditors in each stage t are paid by losing Auditors in stage $\mathsf{t}+1$. In the last stage $\mathsf{T}+1$, the winning Auditors are paid by the PA-deposited bounty. Notably, Agents are not eliminated across stages. They can submit as often as they like, and win as many times as they perform. The diagram continues in fig 2.2. See Appendix D for simplified version of sequential parimutuel model markets.

The Agents' winnings do not come from the PA's bounty, but rather from the losing Auditors' wagers. The Auditors' winnings also stem from other Auditors, except for the Auditors who wagered on stage $\mathsf{T}=60$, in which case their winning come from the bounty $\mathfrak{v}_{\mathsf{pa}}$.

4. Auditors approach the market sequentially in every stage t, and wager on the *previous* stage's submissions (that is $\mathsf{t}-1$). They may wager according to QAudit-protocol's *prompted rankings* over Agents' models, or the Auditors may report their own ranking. QAudit provides nonfungible-tokens for every $\mathsf{Agent}_{\mathsf{k}}$'s model at every time stage, each nonfungible token is denoted $\mathfrak{mint}^{\mathsf{t}}_{\mathsf{k}}$. Each $\mathfrak{mint}^{\mathsf{t}}_{\mathsf{k}}$ token represents the quality of $\mathsf{Agent}_{\mathsf{k}}$'s model in a particular stage. For example in the $\mathsf{QAudit}(60, 24, 100, 3, \mathfrak{v}_{\mathsf{pa}})$ instance, there are 100 Agents participating, each Agent submits one model per day. If the average number of tokens sold per model is 10 $\mathfrak{mint}$s. Then a total of $10 \times 100 \times 60 = 60,000$ $\mathfrak{mint}$ tokens will be sold over the course of 60 days. Specifically, if Auditors have purchased 20 $\mathfrak{mint}^{12}_4$ and 10 $\mathfrak{mint}^{12}_1$ tokens on stage $\mathsf{t}=12$, then QAudit interprets this buy-signal as stating that Agent_4's submission is twice as likely to be the best model as that of Agent_1's submission on the 12th day.

5. Critically, every QAudit instance runs an automated market maker (Amm) so that each time the Auditor wagers by buying $\mathfrak{mint}$ tokens, the price of model shares change in accordance to the Auditor's wager. When the market closes after 24 hours, the 3 winners of the day are chosen as the top 3 Agents most likely to win per market consensus. However, this *does not* imply the top winning Agents or the associated Auditors will be paid immediately. It simply means the top R Agents will be judged against the previous stage's top R teams at test time after 60 days. The Amm resets the price on every stage t, so Agent models' closing price on one stage does not affect the opening price in the next stage.

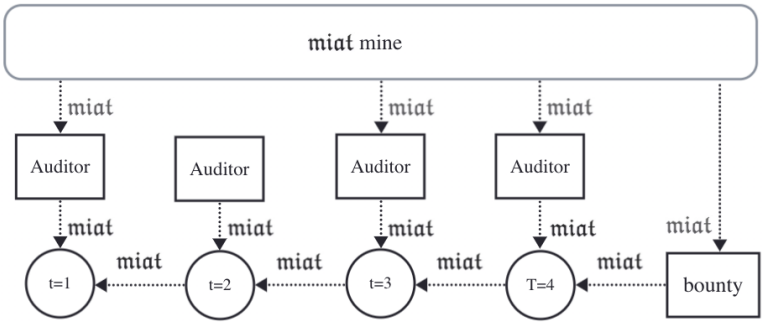


Figure 2.2: In QAudit sequential parimutuel market, Auditors wager with on-chain currency miat that is sold at the current miat:Fat:fiat exchange rate. Upon each purchase event, new miat is minted for Auditors. The second Auditor in the diagram has enough miat to wager, so no additional token is mined. Bounty miat always come from mines. See section 2.2 to understand how QAudit achieves incentive compatibility and budget balance.

6. At competition close after $\mathsf{T}+1$ stages, the test set is revealed and all the submissions are judged against it. In $\mathsf{QAudit}(60, 24, 100, 3, \mathfrak{v}_{\mathsf{pa}})$, the top 3 submissions at any given day is judged against the top 3 submissions in the previous day *collectively*. Specifically, QAudit builds an collective model from the 3 individual submissions and judge this ensemble, so that:

   a) if the ensemble model of the day beats the previous day's ensemble on the test set, then the Agents who submitted the top models are rewarded *individually* from the losing wagers of that day, in proportion the marginal information value add they provided on the collective model. This is the principle of **judging collectively and rewarding individually**.

   b) The Auditors who wagered on the top model are paid from the top wagers of the *next day*, in proportion to how much they have wagered. This is the principle of **eat what you pay for**. A betting regime whereby each person is paid off by the next person in line is called a *sequential parimutuel market*. [1]

   c) Auditors who wagered on the last day are paid from the PA-provided bounty $\mathfrak{v}_{\mathsf{pa}}$.

Most importantly, QAudit does not engage in staking but rather wagering. So Auditors who bet on the losing models are not paid. Agents who submit losing models lose only time and effort. Observe QAudit is parimutuel for Agents, and sequential parimutuel for Auditors. In order to win:

1. Agents need Auditors to wager generously on their models, so that they may be selected by the protocol to compete against the previous day's top model ensemble.

2. Auditors need Agents to submit good models, so that their wager is not lost in vain.

3. Auditors and Agents are paid from different pools: Agents rake today's losing wagers, while Auditors take tomorrow's winning wagers.

In sum, **Auditors and Agents hunt together but eat separately**. This arrangement of incentives is critical as it breaks the conflict of interest that tend to exist between the Auditors and the audited, that is the Agent. Instead of evading each other, the QAudit sequential game play induces a *Nash Equilibrium*, whereby the Agents and Auditors collude to submit the most performant model possible, and wager to move it to the top of the line.

[1] See [Han02],[Han03],[CV10],[Agr+09],[CA10],[ACV12],[CP07],[GR07],[FPW23],[AK+14],[Pen04],[HW19], and [For+07].

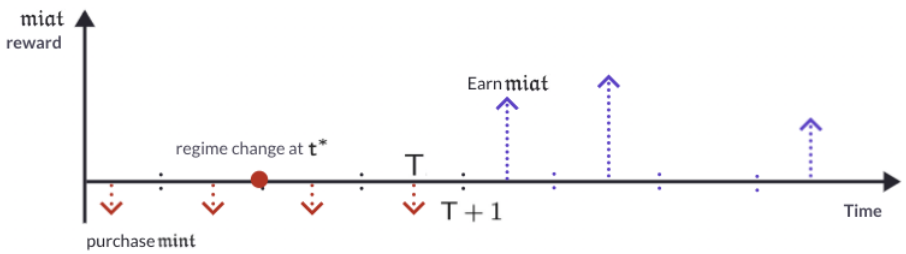


Figure 2.3: QAudit miat-flow diagram. The model/wager elicitation stages lasts from $\mathsf{t} = 1...\mathsf{T}+1$, where Agents submit models, and Auditors buy *machine intellgience nonfungible tokens* (mint) issued from each model. They buy mints with the *machine intellgience audit token*, or miat. The models are scored against a validation data set in a preliminary basis in each stage t. After $\mathsf{T}+1$ stages, QAudit replays the entire market history, and judges the Agent-submitted models on the test set. It allocates miat-denominated winnings to the Agents and Auditors who submitted/wagered on the best models in each stage t. Market participants can win as many times as they submit models or wager, and more than one participant can win each stage. Note the *regime change* at stage t^*, this is an important topic covered in section 2.6.

The introduction above leaves out one last stakeholder: the protocol developer Metheus. The tried-and-true way to profit is with trading fees. We do this whilst putting the settlement layer on chain, so that wagering is passed through three sets of tokens with floating exchange rates: 1) the Fat stable coin, 2) the miat audit token, and 3) the mint machine intelligence nonfungible token. Their floating exchange rate is Metheus's margin opportunity. This is the essence of the system, but the devil is in the subtleties. The rest of the chapter deconstructs the protocol in layers, and proceeds as follows:

1. section 2.2 below tables the protocol incentives at a high level. In particular it introduces the notion of *staggered sequential parimutuel prediction market*. And qualitatively argue that it aligns the incentives of the Auditors and Agents, so that they collude to report and wager the best models.

2. Section 2.3 specify the protocol in detail, and introduces all the terminology and data types that will be used to prove the protocol's properties.

3. Section 2.4 highlight's QAudit's machine learning subroutines. A primer on machine learning may be found in Appendix E.

4. Section 2.5 proves these elementary properties of QAudit:

   a) lemma 2.5.1 shows that the QAudit reward structure is *incentive compatible* from the perspective of the Agents, so that they will report their best model subject to resource constraints. Lemma 2.5.2 then shows that reporting the truth is a *pure strategy equilibrium* for every Agent.

   b) Lemma 2.5.3 states that collusion is a *mixed-strategy equilibrium* between Agent and Auditor. Whereby "collusion" implies the Agents report their best model in every stage t, and the Auditors wager on the best reported model in this stage.

5. Section 2.6 follows the "spirit" of [CV10], and view QAudit as 1) a market protocol, 2) a game, and 3) a randomized algorithm. In this spirit, we state the *wager-complexities* of QAudit - loosely defined as the number of wagers needed for the market to find the

correct ranking $\mathfrak{s}_r^*$ over Agents' submitted models. In particular, we have for every stage $\mathsf{t} = 1 \dots \mathsf{T}+1$:

a) lemma 2.5.8 states that in the event where every $\mathsf{Auditor_i}$ accepts the prompted rankings $\mathfrak{s}_\mathsf{r}^*$ provided by QAudit instance, then the Auditors will collectively find the correct permutation in $\mathsf{poly}(\mathsf{K!})$ number of wagers. Where K is the number of Agents participating in some instance of QAudit market.

b) Lemma 2.5.10 states the "stability" of equilibrium state when the Auditors behave selfishly according to unobserved interest. We find these sensitivity bounds:

i. If some $1-\alpha_\mathsf{r}$ $(0 < \alpha_\mathsf{r} < 1)$ portion of Auditors wager according to unprompted rankings, then the market settles on some distribution $\mathbf{p}$ of over the rankings. It deviate from the delta distribution over the prompted ranking $\tilde{\boldsymbol{\delta}}_\mathsf{r}$ by:

$$\left\|\mathbf{p} - \tilde{\boldsymbol{\delta}}_\mathsf{r}\right\|_1 \leq \frac{2}{1-\alpha_\mathsf{r}}|\Delta\alpha_\mathsf{r}|.$$

ii. The deviation in the QAudit equilibrium as a function of the Auditors' private preference over rankings $(\mathbf{e} + \Delta\mathbf{e})$ is:

$$\|\tilde{\boldsymbol{\delta}}_\mathsf{r} - \mathbf{p}\|_1 \leq (1-\alpha_\mathsf{r})\|\Delta\mathbf{e}\|_1.$$

iii. The sensitivity of distribution on rankings as a function of the $\mathsf{Auditor_r}$'s' wager strategy $\mathbf{M}_\mathsf{r}$ is:

$$\|\tilde{\boldsymbol{\delta}}_\mathsf{r} - \mathbf{p}\|_1 \leq \frac{\alpha_\mathsf{r}}{1-\alpha_\mathsf{r}}\|\Delta\mathbf{M}_\mathsf{r}\|_\infty.$$

c) Lemma 2.5.11 states that in the event where the Auditors are partitioned into two blocks, so that some of the Auditors accept the prompted permutation, denoted $\mathfrak{s}_\mathsf{r}^*$, and some of which hold a different opinion. If the proportion of Auditors who hold private belief $\mathfrak{s}_\mathsf{r}^*$ is greater than 50%, then under some very mild conditions on the rate of wagering, the likelihood of finding one of the two accepted solution states may be significantly greater than $\frac{1}{2}$.

6. Section 2.6 then builds on the previous lemmas, and states *three fundamental theorems of* QAudit. The first theorem captures market behavior when Auditors possess "singular conviction." The second theorem explains the rationality of "herd behavior" in QAudit markets. Notably, both conviction and herd behavior lead to correct rankings in expectation. The third theorem states that despite the intrinsic oscillation of the price bonding curve, QAudit is profitable for winning Auditors.

a) Theorem 2.6.4, or the QAudit *Protocol Consensus Theorem*, stating that for every stage t, there exists a *Correlated Equilibrium*, s.t. the market consensus at market close reflects the true permutation $\mathfrak{s}_\mathsf{r}^{*\mathsf{t}}$ in expectation, provided the permutation is prompted by $\mathsf{QAudit_\mathfrak{v}}$ at market open in every t.

b) Theorem 2.6.6, or the QAudit *Social Consensus Theorem* shows that even if Auditors differ in their belief over the optimal ranking over Agents, rational herd behavior will push the market towards the correct ranking. Specifically, we state that in the event where the protocol suggests many possible rankings over models, then in every protocol stage t, there exists a point in time τ^* bounded by:

$$\tau_f - \mathsf{N} - \mathsf{K}\log(\mathsf{K}) \leq \tau^* < \tau_f - 1 - \mathsf{K}\log(\mathsf{K}),$$

s.t. it is rational for every Auditor to switch to wagering on one unique ranking.

c) Theorem 2.6.10: QAudit *is not a Loser's Game*. That is to say the QAudit sequential parimutuel reward arrangement has positive expected value for winning Auditors. In particular, we find that Auditors profit if the per-stage-t earnings $\Delta\mathfrak{v}$ is above this threshold:

$$\Delta\mathfrak{v} \geq \frac{8\mathfrak{v}_{\mathsf{t}^*}}{\sum_{\mathsf{t}=2}^{\mathsf{t}^*}(1+\curlywedge)^{\mathsf{t}^*-\mathsf{t}}}.$$

Where:

i. $\curlywedge$ is *regime change risk* as seen in definition 2.6.9.
ii. $\mathfrak{v}_{\mathsf{t}^*}$ is the Auditor$_\mathsf{i}$'s wager at the moment of regime change t^*.
iii. the interval $[\mathsf{t}, \mathsf{t}^*]$ defines the stages between consecutive regime changes.

7. Section 2.7 introduces *perpetual* QAudit, where the basic QAudit market is extended so that the game has infinite time horizons. Then it shows how this game can recreate an asset curve over mint tokens that is comparable to Bitcoin. Featuring periodic price oscillations similar to Bitcoin halving events, and an over-all upward drift in price ceiling over time.

2.2 QAudit Incentive Structure

This section offers a high-level explanation of the incentive structure of the QAudit automated market maker (Amm). It begins by reviewing a basic sequential prediction market, and use its deficiency to motivate the QAudit price-bonding curve. At a very basic level, QAudit is an information market. They are similar to stock markets, but emphasize 1) incentive compatible pricing rules, or proper scoring rules; and 2) budget balance. A *proper scoring rule*:

1. elicits the truthful report of private information from agents,
2. collate all the reports a probability distribution that reflects market consensus,
3. and compensates the agents in accordance to the quality of their report.

Reports in information markets are always judged against some objective ground truth at the end of the competition. For example, if some information market is designed to elicit the likelihood of raining next Tuesday, then the players are judged by whether it does rain on Tuesday. Thus unlike stock markets where price can be conflated for value and traders are rewarded for pumping the stock, in information markets the users are rewarded for telling the truth. Moreover the QAudit Amm is *budget balanced*:

1. the current stage's winners are paid by the next stages' wagers.
2. The last tranche of players to participate in the QAudit market is paid by the bounty paid by the PA.

Thus neither the PAs nor Metheus will go bankrupt operating QAudit protocols. See Appendix D for more on sequential parimutuel information markets. Presently, using prediction market to run collaborative machine learning projects has been proposed by the Ethereum Foundation in [HW19], and fielded at Amazon with [GH+24]. QAudit is similar to these proposed sequential parimutuel markets in appearance, but differs in some important ways:

1. typical prediction markets do not differentiate between principles who make the bets, and Agent who report the information without paying, in fact they may be paid for reporting good models. QAudit separates Agent who submit the model checkpoints, and Auditor who wager on the models. [2] This segregates the two classes of users who have different relationship with technology. Most importantly, Agent-Auditor dichotomy

[2] A checkpoint is the intermediate state of a model during training. They are useful for diagnostics.

ensures the machine learning scientists do not have to "pay-to-play," which is unnatural from their perspective to put it charitably. However, separation of reporting and profiting introduces principle-agent conflict of interests, as Agents are no longer directly rewarded or punished for their submissions. In the main body, we argue QAudit induces an equilibrium so that they *do* collude to report and wager on the best models.

Agent reports model $\boldsymbol{\theta}$ or $\mathbb{D}$, Auditor wagers on reports

Player	Report	Amm update
$\mathsf{Agent}_{\mathsf{k}}$	$\boldsymbol{\theta}_{\mathsf{k}}^{\mathsf{t}}$	$\boldsymbol{\theta}_{\mathsf{k}}^{\mathsf{t}} \to \mathbf{w}_{\mathsf{k}}^{\mathsf{t}} \to \log \mathsf{p}_{\mathsf{k}}^{\mathsf{t}}$
$\mathsf{Agent}_{\mathsf{k}}$	$\mathbb{D}_{\mathsf{k}}^{\mathsf{t}}$	$\mathbb{D}_{\mathsf{k}}^{\mathsf{t}} \to \mathbf{w}_{\mathsf{k}}^{\mathsf{t}} \to \log \mathsf{p}_{\mathsf{k}}^{\mathsf{t}}$
$\mathsf{Auditor}_i$	$\mathbf{b}_i^{\mathsf{t}} = \left(\mathsf{b}_{1,\mathsf{i}}^{\mathsf{t}}, \ldots, \mathsf{b}_{\mathsf{K},\mathsf{i}}^{\mathsf{t}}\right)$	$\mathbf{s}^{\mathsf{t}} = \mathbf{s}^{\mathsf{t}} + \mathbf{b}_i^{\mathsf{t}}$

Table 2.1: In QAudit, Agents report either their generative model $\boldsymbol{\theta}$ or raw data set $\mathbb{D}$. Auditors report a buy vector $\mathbf{b}$, where each scalar $\mathsf{b}_{\mathsf{k},\mathsf{i}}^{\mathsf{t}}$ denotes how many shares of model k $\mathsf{Auditor}_{\mathsf{i}}$ will buy. The $\mathsf{QAudit}_{\mathsf{Amm}}$ then trains a logistic regression classifier $\mathfrak{p}(\mathsf{y}|\mathbf{x}; \mathbf{w}_{\mathsf{k}}^{\mathsf{t}}) = \boldsymbol{\sigma}(\langle \mathbf{w}_{\mathsf{k}}^{\mathsf{t}}, \mathbf{x} \rangle)$ by sampling from either the reported data set $\mathbb{D}$, or the generative model $\mathfrak{p}(\mathbf{x}_\tau|\mathbf{x}_{\tau-1}; \boldsymbol{\theta})$. This model outputs a likelihood $\mathsf{p}_{\mathsf{k}}^{\mathsf{t}}$ that scores the Agent's reported model $\boldsymbol{\theta}$. Similarly, the QAudit protocol updates the market inventory with the Auditor's orders when he reports a buy vector. Here $\mathbf{s}^{\mathsf{t}} + \mathbf{b}_i^{\mathsf{t}}$ is element-wise vector addition.

2. Classical formulation of prediction markets for machine learning models assume the agents are all working on the same underlying model. They update the weights in private before submitting the candidate updates to a shared global model state [[HW19], [GH+24]]. This assumption is very limiting and defeats the purpose of holding a competition, which is to explore different solutions. In normal machine learning competition, teams work on separate repositories, and often use different model architectures. QAudit extends the sequential prediction market design pattern to accommodate a more natural use case whereby the teams share the same objective, but select their own model architectures. In fact, QAudit can be used even if teams do not submit models at all.

3. Proper market scoring rules must be convex functions, so the naive formulation of machine learning sequential prediction markets requires the model to be drawn from the exponential family [[GR07], [HW19], [AK+14]]. For example, logistic regression satisfies this constraint (see Appendix E.3.3, and Appendix F for more). However this is a highly contrived setting, there is no need to crowd source regression. Instead QAudit aims for production level assets. In particular, the protocol elicits two types of products from Agents:

   a) (*generative models*): of form $\mathfrak{p}(\mathbf{x}_\tau|\mathbf{x}_{\tau-1}; \boldsymbol{\theta})$, where $\mathbf{x}_{\tau-1}$ is the query and $\mathbf{x}_\tau$ is the response. These deep artificial neural networks (ANNs) do not have convex partition functions. Moreover, most commercially interesting ANNs such as large language models (LLMs) and image generation models do not output a probability at all, instead they output real values. Thus QAudit trains a surrogate classifier $\mathfrak{p}(\mathsf{y}|\mathbf{x}; \mathbf{w})$ on responses sampled from generative models $\boldsymbol{\theta}$, and some noise control $\boldsymbol{\eta}$ or natural data set $\mathbb{D}$. The classifier learns to discriminate noise from generated sample, or generated sample vs real data set. This fraud-detection game is somewhat similar to generative adversarial nets of [GP+14]. However in our case the objective functions are not tied, and the QAudit protocol is interested in the discriminative model, not the generator.

   b) (*raw data*): similar to the previous point, raw data is not equipped with a convex log partition function. But it can be used to train a convex surrogate for scoring. A QAudit market instance that is eliciting data from agents is in fact a *data market*,

the PA and Auditors are buying information from users. This is an important aspect of the Metheus economy. Data markets significantly expand the surface area of who can participate in the games. Most people cannot train machine learning models, but most people do have access to some data that can be sold.

In both cases, if the Agent-submission is of high quality, then it would refine the decision boundary between the source and the control learned by the classifier $\mathbf{w}$. If the submission is low quality, then it degrade this boundary. This is reflected in changing classification error w.r.t. the test set provided by the principle auditor PA. QAudit measures the improvement or degradation in quality of the classifier, and uses it to reward the Agents and Auditors.

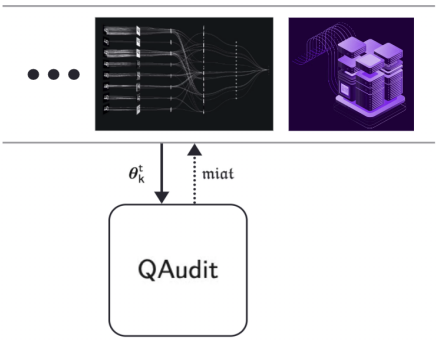


Figure 2.4: QAudit accepts two forms of submissions from Agents: 1) machine learning models $\boldsymbol{\theta}$ (such as LLM), which accepts a query and outputs reponse with $\mathfrak{p}(\phi_{\mathbf{x}_\tau}|\phi_{\mathbf{x}_{\tau-1}};\boldsymbol{\theta})$; or 2) raw data sets $\mathbb{D}$ that can be used to train models downstream. In the first case, QAudit in a model market, in the second case, it is a data market.

4. However scoring a convex surrogate introduces a problem: QAudit must train the surrogate classifier by sampling from the reported models first, and then publicize the price. Hence the QAudit market refactors auditing into two steps: *model elicitation step* and *wager elicitation step*. After all the generative models or datasets have been reported, QAudit runs discriminative model on the reported assets and then publish the relative performance of these discriminative models proxies.

5. Additionally, splitting the market into two steps bottle necks the trading or auditing activity, so that the Auditors have a full stage (i.e. 24 hours) to trade in and out of all K models. If we allow the Agents to submit whenever they "feel like it" as is the case with [HW19], then some agents may game the system by submitting early and often. This skews the incentive and turns the auditing process into a spam-like popularity contest. Note the two steps are interleaved, so that i.e. if one time stage is 24 hours, then the Auditors are always trading yesterday's models.

6. In the classical formulation,the price charged is computed in expectation over data samples, whilst the reward is computed on a particular data set. Should there be any biases in the sample, then there is a mismatch between wagers collected and rewards given. The market maker must then make up the difference - a prohibitive ask.

In sum, QAudit differs from standard prediction markets in that we 1) consider a wider set of stakeholders, 2) think deeply about the consequence of protocol design on user experience, and 3) the protocol accept both models with non-convex partition functions, and datasets with no partition functions at all. Finally, note that the auditors in QAudit are not staking but rather *wagering*. Staking implies the principle is returned, while wagering make no such guarantees. Lost wagers will be used to subsidize cost of training for Agents. This is is an important point: **trading pays for training**.

QAudit Staggered Sequential Parimutuel AMM

QAudit builds upon previous work in sequential parimutuel prediction markets. However we must solve two problems:

1. the Auditors are not staking but wagering. And since Auditors are paid from the next stages' wagers, an amount of which he cannot control, it is possible for a Auditor to lose money even if he selects the correct model, but overpays for the current one.

2. Since Agents are not staking on their own submission but rely on Auditors to pick them, QAudit must 1) incentivize the Agents to report the best model they can, and 2) prevent the Auditors from selecting a lesser model over a better one.

The next three points builds towards the full QAudit incentive structure that remedy the principle-agent problems.

1. (QAudit *simple*) In this simplified version, Agents submit their models $\boldsymbol{\theta}$ at every stage t, while Auditors wager on the submitted models from each stage $\mathsf{t}-1$. Then during the reward phase, QAudit allocate the rewards depending on how well the top Agents selected by the Auditors have performed. The problem is that Auditors may wager as much as they want per stage, with no *price floor* on the mint model shares. So that Auditors may lose money even if they win the stage. For example referencing table 2.2, at $\mathsf{t}=1$ $\mathsf{Auditor}_1$ wins the stage by wagering $\mathfrak{v}^1_{\mathsf{alice},1}$. But he does not control how much the winning Auditors at stage $\mathsf{t}=2$ wagers. So that if $\mathfrak{v}^1_{\mathsf{alice},1} \geq \mathfrak{v}^2_{\mathsf{alice},2} + \mathfrak{v}^2_{\mathsf{alice},3}$, then $\mathsf{Auditor}_1$ may still lose money.

QAudit: Agent reports $\boldsymbol{\theta}$, Auditor wagers with no price floor

Stage t	Auditor wagers $\mathfrak{v}^{\mathsf{t}}_{\mathsf{k,i}}$	Agent reports $\boldsymbol{\theta}$	reward $\mathfrak{v}^{\mathsf{t}}_{\mathsf{Agent}}$	reward $\mathfrak{v}^{\mathsf{t}}_{\mathsf{Auditor}}$
0	0	$\boldsymbol{\theta}^0_{\mathsf{Amm}} \to \log 0.5$	0	0
1	$\mathfrak{v}^1_{\mathsf{alice}}$ $\mathfrak{v}^1_{\mathsf{bob},2} + \mathfrak{v}^1_{\mathsf{bob},3}$	$\boldsymbol{\theta}^1_{\mathsf{alice}} \to \log 0.6$ $\boldsymbol{\theta}^1_{\mathsf{bob}} \to \log 0.3$	$\mathfrak{v}^1_{\mathsf{bob},2} + \mathfrak{v}^1_{\mathsf{bob},3}$ 0	$\mathfrak{v}^2_{\mathsf{alice},2} + \mathfrak{v}^2_{\mathsf{alice},3}$ 0
2	$\mathfrak{v}^2_{\mathsf{alice},2} + \mathfrak{v}^2_{\mathsf{alice},3}$ $\mathfrak{v}^2_{\mathsf{eve},3}$	$\boldsymbol{\theta}^2_{\mathsf{alice}} \to \log 0.7$ $\boldsymbol{\theta}^2_{\mathsf{eve}} \to \log 0.5$	$\mathfrak{v}^2_{\mathsf{eve},3}$ 0	$\mathfrak{v}^3_{\mathsf{bob},1} + \mathfrak{v}^3_{\mathsf{bob},2}$ 0
$\mathsf{T}=3$	$\mathfrak{v}^3_{\mathsf{bob},1} + \mathfrak{v}^3_{\mathsf{bob},2}$ $\mathfrak{v}^3_{\mathsf{eve},3}$	$\boldsymbol{\theta}^3_{\mathsf{bob}} \to \log 0.8$ $\boldsymbol{\theta}^3_{\mathsf{eve}} \to \log 0.9$	$\mathfrak{v}^3_{\mathsf{eve},3}$ 0	bounty $\mathfrak{v}_{\mathsf{pa}}$ 0

Table 2.2: A simple audit example where $\mathsf{T}=3$. The Agents report model $\boldsymbol{\theta}$ used to train classifier $\mathbf{w}$ (not shown here). QAudit then judges the quality of model on test data.In this case a higher probability value in $\log \mathsf{p}$ is a better estimate. Note $\mathsf{T}=3$, so the last Auditor who wagers correctly is rewarded with the bounty value. This is called budget balance since the QAudit Amm is not paying for any of the reports.

2. (QAudit *with Dutch auction*) The solution is to vary the price floor in every stage with the function $\mathsf{v}^{\mathsf{t}}_{\rho}$. So that we can maintain some invariant of form:

$$\mathsf{v}^1_{\rho} \cdot \mathfrak{v}^1_{\mathsf{alice},1} \leq \mathsf{v}^2_{\rho}\big(\mathfrak{v}^2_{\mathsf{alice},2} + \mathfrak{v}^2_{\mathsf{alice},3}\big),$$

if the shares "sell well" over the stages. And the inverse inequality:

$$\mathsf{v}^1_{\rho} \cdot \mathfrak{v}^1_{\mathsf{alice},1} > \mathsf{v}^2_{\rho}\big(\mathfrak{v}^2_{\mathsf{alice},2} + \mathfrak{v}^2_{\mathsf{alice},3}\big),$$

in the event where the price is too high so the shares fail to sell. This reversion in price is known as *regime change* (see section 2.6). The Auditor is effectively evaluating an asset $\boldsymbol{\theta}^{\mathsf{t}}_{\mathsf{k}}$, whose performance or fundamental value is increasing every stage with upward drift in price. He must decide when and how often to buy $\mathsf{mint}^{\mathsf{t}}_{k}$ shares of $\boldsymbol{\theta}^{\mathsf{t}}_{\mathsf{k}}$. Naively the Auditor should buy in one of the earlier stages when bonding factor $\mathsf{v}^{\mathsf{t}}_{\rho}$ is a lower value.

QAudit: Agent reports $\boldsymbol{\theta}$, Auditor wagers with variable price floor

Stage t	Auditor wagers $\mathfrak{v}^{\mathsf{t}}_{\mathsf{k,i}}$	Agent reports $\boldsymbol{\theta}$	reward $\mathfrak{v}^{\mathsf{t}}_{\mathsf{Agent}}$	reward $\mathfrak{v}^{\mathsf{t}}_{\mathsf{Auditor}}$
0	0	$\boldsymbol{\theta}^0_{\mathsf{Amm}} \to \log 0.5$	0	0
1	$\mathsf{v}^1_\rho \cdot \mathfrak{v}^1_{\mathsf{alice},1}$ $\mathsf{v}^1_\rho(\mathfrak{v}^1_{\mathsf{bob},2}, \mathfrak{v}^1_{\mathsf{bob},3})$	$\boldsymbol{\theta}^1_{\mathsf{alice}} \to \log 0.6$ $\boldsymbol{\theta}^1_{\mathsf{bob}} \to \log 0.3$	$\mathsf{v}^1_\rho(\mathfrak{v}^1_{\mathsf{bob},2} + \mathfrak{v}^1_{\mathsf{bob},3})$ 0	$\mathsf{v}^2_\rho(\mathfrak{v}^2_{\mathsf{alice},2} + \mathfrak{v}^2_{\mathsf{alice},3})$ 0
2	$\mathsf{v}^2_\rho(\mathfrak{v}^2_{\mathsf{alice},2}, \mathfrak{v}^2_{\mathsf{alice},3})$ $\mathsf{v}^2_\rho \cdot \mathfrak{v}^2_{\mathsf{eve},3}$	$\boldsymbol{\theta}^2_{\mathsf{alice}} \to \log 0.7$ $\boldsymbol{\theta}^2_{\mathsf{eve}} \to \log 0.5$	$\mathsf{v}^2_\rho \cdot \mathfrak{v}^2_{\mathsf{eve},3}$ 0	$\mathsf{v}^3_\rho(\mathfrak{v}^3_{\mathsf{bob},1} + \mathfrak{v}^3_{\mathsf{bob},2})$ 0

Table 2.3: In QAudit with ascending price factor, the Amm scales the price computed by the logarithmic market scoring rule by factor $\mathsf{v}^{\mathsf{t}}_\rho$. However the QAudit Amm does not increase the price monotonically across stages $\mathsf{t} = 1..\mathsf{T}$, this would eventually price all buyers out of the market. Instead, QAudit features periodic *regime changes*, whereby the price is reset to some initial point. See 2.6.8 for more. Last row omitted in table above.

QAudit: Agent reports $\boldsymbol{\theta}$, Auditor wagers with Dutch-auction-set price

Stage t	Auditor wagers $\mathfrak{v}^{\mathsf{t}}_{\mathsf{k,i}}$	Agent reports $\boldsymbol{\theta}$	reward $\mathfrak{v}^{\mathsf{t}}_{\mathsf{Agent}}$	reward $\mathfrak{v}^{\mathsf{t}}_{\mathsf{Auditor}}$
1	v^1_ρ $(\mathsf{v}^1_\rho, \mathsf{v}^1_\rho)$	$\boldsymbol{\theta}^1_{\mathsf{alice}} \to \log 0.6$ $\boldsymbol{\theta}^1_{\mathsf{bob}} \to \log 0.3$	$2\mathsf{v}^1_\rho$ 0	$2\mathsf{v}^2_\rho$ 0

Table 2.4: Simplified $\mathsf{t} = 1$ betting history. The Auditors' wagers are set to $\mathfrak{v}^{\mathsf{t}}_{\mathsf{k,i}} = 1$, so they bid the price fixed by bonding function $\mathsf{v}^{\mathsf{t}}_\rho$. The other stages are omitted for clarity.

3. (QAudit *full*) There is still a problem with table 2.2 in time stage $\mathsf{t} = 3$: note that even though $\boldsymbol{\theta}^3_{\mathsf{eve}}$ outperformed $\boldsymbol{\theta}^3_{\mathsf{bob}}$, $\mathsf{Auditor}_1$ and $\mathsf{Auditor}_2$ have wagered on $\mathsf{Agent}_{\mathsf{bob}}$ instead, thus $\mathsf{Agent}_{\mathsf{eve}}$ lost unfairly. The solution is to penalize the Auditors with regret factor ϱ^{t}_k, and allocate this regret to eve. *Regret* expresses how well the ensemble model could have done had the Auditors selected the best models. It prevents the Auditors from defecting from the best model on each stage, and prompts the Agents to submit their best model knowing they will be rewarded by either the Auditors, or the QAudit regret compensation. The exact value of this regret depends on how well $\boldsymbol{\theta}^3_{\mathsf{eve}}$ does, see table below and subsection 2.3 for more information.

QAudit: Agent reports $\boldsymbol{\theta}$, Auditor wagers, variable price floor, regret penalty

Stage t	Auditor wagers $\mathfrak{v}^{\mathsf{t}}_{\mathsf{k,i}}$	Agent reports $\boldsymbol{\theta}$	reward $\mathfrak{v}^{\mathsf{t}}_{\mathsf{Agent}}$	reward $\mathfrak{v}^{\mathsf{t}}_{\mathsf{Auditor}}$
0	0	$\boldsymbol{\theta}^0_{\mathsf{Amm}} \to \log 0.5$	0	0
1	$\mathsf{v}^1_\rho \cdot \mathfrak{v}^1_{\mathsf{alice},1}$ $\mathsf{v}^1_\rho(\mathfrak{v}^1_{\mathsf{bob},2}, \mathfrak{v}^1_{\mathsf{bob},3})$	$\boldsymbol{\theta}^1_{\mathsf{alice}} \to \log 0.6$ $\boldsymbol{\theta}^1_{\mathsf{bob}} \to \log 0.3$	$\mathsf{v}^1_\rho(\mathfrak{v}^1_{\mathsf{bob},2} + \mathfrak{v}^1_{\mathsf{bob},3})$ 0	$\mathsf{v}^2_\rho(\mathfrak{v}^2_{\mathsf{alice},2} + \mathfrak{v}^2_{\mathsf{alice},3})$ 0
2	$\mathsf{v}^2_\rho(\mathfrak{v}^2_{\mathsf{alice},2}, \mathfrak{v}^2_{\mathsf{alice},3})$ $\mathsf{v}^2_\rho \cdot \mathfrak{v}^2_{\mathsf{eve},3}$	$\boldsymbol{\theta}^2_{\mathsf{alice}} \to \log 0.7$ $\boldsymbol{\theta}^2_{\mathsf{eve}} \to \log 0.5$	$\mathsf{v}^2_\rho \cdot \mathfrak{v}^2_{\mathsf{eve},3}$ 0	$\mathsf{v}^3_\rho(\mathfrak{v}^3_{\mathsf{bob},1} + \mathfrak{v}^3_{\mathsf{bob},2})$ 0
$\mathsf{T} = 3$	$\mathsf{v}^3_\rho(\mathfrak{v}^3_{\mathsf{bob},1}, \mathfrak{v}^3_{\mathsf{bob},2})$ $\mathsf{v}^3_\rho \cdot \mathfrak{v}^3_{\mathsf{eve},3}$	$\boldsymbol{\theta}^3_{\mathsf{bob}} \to \log \mathsf{p}^3_{\mathsf{bob}}$ $\boldsymbol{\theta}^3_{\mathsf{eve}} \to \log \mathsf{p}^3_{\mathsf{eve}}$	$\mathsf{v}^3_\rho \cdot \mathfrak{v}^3_{\mathsf{eve},3}$ ϱ^3_{eve}	$\mathfrak{v}_{\mathsf{pa}} - \varrho^3_{\mathsf{eve}}$ 0

Table 2.5: Full incentive structure of QAudit, featuring parimutuel payout per stage for Agents, and sequential parimutuel payouts for Auditors, featuring Dutch ascending auction factor v_ρ and regret ϱ penalty.

2.3 QAudit Core

This section presents QAudit, highlighting how Agents, Auditors and $\mathfrak{Fat}$ holders can profit. The details of the protocol are factored into multiple subroutines. In the future, certain subroutines may compose with open source variations. QAudit is subtle, it introduces:

1. Auditor-Agent dichotomy.
2. It separates model elicitation step from wager elicitation step.
3. And converts amongst three tokens ($\mathfrak{Fat}, \mathfrak{miat}, \mathfrak{mint}$) with variable exchange rates.

QAudit differs dramatically from the "straight-forward," but naive solutions of vanilla sequential-parimutuel machine learning markets of Appendix D.2. Aside from fiat, the following set of commodities are "in play" for every QAudit instance:

1. ($\mathfrak{Fat}$): the fully-backed stable coin pegged to 1 fiat of choice. When Auditors buy into the market, they must purchase $\mathfrak{miat}$ token using fiat. This purchase is then exchanged to $\mathfrak{miat}$ at the current floating $\mathfrak{miat}$ to $\mathfrak{Fat}$ exchange ratio. The $\mathfrak{miat}$ may come from a pre-mined $\mathfrak{miat}$ pool, or if the pool is empty then $\mathfrak{miat}$ will be minted for conversion. This conversion occurs in the backstage and do not require user decisions.

2. ($\mathfrak{miat}$): the machine intelligence audit token. Auditors use $\mathfrak{miat}$ to purchase model shares over stages $\mathsf{t} = 1...\mathsf{T}+1$, and are rewarded in $\mathfrak{miat}$ if their models win. This $\mathfrak{miat}$ is then convertible back to $\mathfrak{Fat}$ at a floating exchange rate, and at user discretion from $\mathfrak{Fat}$ to fiat at 1:1 basis.

3. ($\mathfrak{mint}_k^{\mathsf{t}}$): the machine intelligence nonfungible token for model k indexed by stage t, or simply "model shares." Model shares express the upside the Auditor is entitled to in the event where model k in stage t is performant relative to previous stage $\mathsf{t}-1$. Thus $\mathfrak{mint}_{\mathsf{k}}^{\mathsf{t}}$ is a claim on the next stage's wager. Once purchased, the Auditor may resell $\mathfrak{mint}$ at their own discretion. Note however, since QAudit allocates the losing Auditor's wagers to the winning Agents at every stage, $\mathfrak{mint}_{\mathsf{k}}^{\mathsf{t}}$ only has value *if* model $\boldsymbol{\theta}_{\mathsf{k}}^{\mathsf{t}}$ wins its stage. [3] The $\mathfrak{mint}_{\mathsf{k}}^{\mathsf{t}}$ is effectively an out of the money call option purchased at time t for model k. It matures at the end of the audit stage $\mathsf{T}+1$, and represents a contingent claim on the wagers of stage $\mathsf{t}+1$. The fact that this option is liquid induces a secondary market on $\mathfrak{mint}$'s that further enriches the QAudit ecosystem.

4. (*data source* $\boldsymbol{\theta}$ *or* $\mathbb{D}$): $\boldsymbol{\theta}$ denotes some generative model, while $\mathbb{D}$ is a data set drawn from nature. This $\mathbb{D}$ may emit data such as real images, videos, and text.

5. (*surrogate classifier* $\mathbf{w}$): each QAudit market keeps a set of latent states, whereby at any stage t, the protocol trains a surrogate proxy of every model $\boldsymbol{\theta}_{\mathsf{k}}^{\mathsf{t}}$ or data set $\mathbb{D}_{\mathsf{k}}^{\mathsf{t}}$ to construct: $\left\{\{\mathbf{w}_1^1, \dots \mathbf{w}_{\mathsf{K}}^1\}, \dots, \{\mathbf{w}_1^{\mathsf{T}}, \dots \mathbf{w}_{\mathsf{K}}^{\mathsf{T}}\}\right\}$ over the course of the full audit session.

Require Statements

Before competition commences, the principle auditor (PA) defines:

1. A fiat-denominated bounty $\mathfrak{v}_{\mathsf{pa}}$ is charged to the principle auditor using the function $\mathsf{QAudit}_{\mathfrak{v}_{\mathsf{pa}}}$. This bounty is allocated to the final set of Agents who submit high quality models as mining reward.

2. *Competition duration* T *over stages* $\mathsf{t} \in \{1, .., \mathsf{T}+1\}$. For example, T may be 60 days where each stage t is 24 hours. On day or stage 0, QAudit initiates the submissions with a randomly initialized model $\boldsymbol{\theta}^o$ and surrogate classifier $\mathbf{w}^o$. See Appendix E for more on deep learning and classical learning methods. Then on day 1, the market opens and accept model submissions in private for stage $\mathsf{t} = 1$.

[3] A nonfungible token is the bar-code for some digital merchandize.

Protocol 1 Machine Intelligence Quality Audit Protocol (QAudit)

Require: Bounty $\mathfrak{v}_{\mathsf{pa}}$, training stage T, qualifiers per stage R
Require: hidden data set $\mathbb{D}$, public query set $\mathbb{Q}$, kernel embedding function ϕ,
Require: noise generator $\boldsymbol{\eta}$, dutch auction function $\mathsf{v}^{\mathsf{t}}_{\rho}$, permutation distance $\mathsf{a} \mid\mid \mathsf{b}$
Require: agent accounts $\{\mathsf{Agent}_{\mathsf{k}}\}_{\mathsf{k}=1..\mathsf{K}}$, and auditor accounts $\{\mathsf{Auditor}_{\mathsf{i}}\}_{\mathsf{i}=1..\mathsf{N}}$

commitment phase:
$(\mathbb{D}_{\mathsf{test}}, \mathbb{D}_{\mathsf{val}}) \leftarrow \mathsf{QAudit}_{\mathbb{D}}(\mathbb{D}, \boldsymbol{\eta}, \phi)$
QAudit reveal validation set $\mathbb{D}_{\mathsf{val}}$

audit phase: ▷ *accept submissions/wagers on models*
$(\boldsymbol{\theta}^{o}, \mathbf{w}^{o}) \leftarrow (\epsilon\mathbf{I}, \epsilon\mathbf{I})$
$(\Theta, \mathbf{W}, \mathsf{logprob}) \leftarrow \left(\left[\{\boldsymbol{\theta}^{o}_{\mathsf{k}}\}_{\mathsf{K}}\right], \left[\{\mathbf{w}^{o}_{\mathsf{k}}\}_{\mathsf{K}}\right], []\right)$
$(\mathbf{B}, \mathbf{P}) \leftarrow (\{\}, \{\})$
for $\mathsf{t} \in 1 \ldots \mathsf{T}+1$ **do**
  if $\mathsf{t} \leq \mathsf{T}$ **then** ▷ *model elicitation phase*
    $(\Theta^{\mathsf{t}}, \mathbf{W}^{\mathsf{t}}, \mathsf{logprob}^{\mathsf{t}}) \leftarrow \mathsf{QAudit}_{\Theta}\left(\mathbb{D}_{\mathsf{val}}, \mathbb{Q}, \boldsymbol{\eta}, \{\mathbf{w}^{\mathsf{t}-1}_{\mathsf{k}}\}_{\mathsf{K}}, \{\mathsf{Agent}_{\mathsf{k}}\}_{\mathsf{K}}\right)$
    $(\Theta, \mathbf{W}, \mathsf{logprob}) \leftarrow ([..., \Theta^{\mathsf{t}}], [..., \mathbf{W}^{\mathsf{t}}], [..., \mathsf{logprob}^{\mathsf{t}}])$
  end if
  if $\mathsf{t} \geq 2$ **then** ▷ *wager elicitation phase with* Amm
    $\left(\mathbf{B}^{\mathsf{t}}, \boldsymbol{\rho}^{\mathsf{t}}\right) \leftarrow \mathsf{QAudit}_{\mathfrak{v}}\left(\mathsf{v}_{\rho}, \mathsf{v}^{o}_{\rho},\ \mathbb{D}_{\mathsf{val}}, \mathbf{W}^{\mathsf{t}-1}, \mathbf{W}^{\mathsf{t}}, \mathsf{logprob}, \{\mathsf{Auditor}_{\mathsf{i}}\}_{\mathsf{N}}\right)$
    $(\mathbf{B}, \mathbf{P}) \leftarrow \left([..., \mathbf{B}^{\mathsf{t}}], [..., \boldsymbol{\rho}^{\mathsf{t}}]\right)$
  end if
end for

reward phase: ▷ *replay game history and allocate reward*
$\mathbf{V} \leftarrow \{\}$
$\left(\Delta, \Delta_{\mathsf{Agent}}, \Delta_{\mathsf{Auditor}}\right) \leftarrow \left([], \{\}, \{\}\right)$
for $\mathsf{t} \in 1 \ldots \mathsf{T}$ **do**
  $(\varrho^{\mathsf{t}}, ...) \leftarrow \mathsf{QAudit}_{\varrho}(\mathsf{a} \mid\mid \mathsf{b}, \mathsf{R}, \mathbf{W}^{\mathsf{t}}, \boldsymbol{\rho}^{\mathsf{t}}, \mathbb{D}_{\mathsf{test}}, \mathbb{D}_{\mathsf{val}})$
  $(\mathsf{logprob}^{\mathsf{t}}_{\mathbb{D}_{\mathsf{test}}}, ...) \leftarrow \mathsf{QAudit}_{\mathsf{infer}}\left(\mathsf{R}, \mathbb{D}_{\mathsf{test}}, \boldsymbol{\rho}^{\mathsf{t}}, \mathbf{W}^{\mathsf{t}}\right)$
  $(\mathsf{logprob}^{\mathsf{t}-1}_{\mathbb{D}_{\mathsf{test}}}, ...) \leftarrow \mathsf{QAudit}_{\mathsf{infer}}\left(\mathsf{R}, \mathbb{D}_{\mathsf{test}}, \boldsymbol{\rho}^{\mathsf{t}-1}, \mathbf{W}^{\mathsf{t}-1}\right)$
  if $\exp(\mathsf{logprob}^{\mathsf{t}}_{\mathbb{D}_{\mathsf{test}}}) > \exp(\mathsf{logprob}^{\mathsf{t}-1}_{\mathbb{D}_{\mathsf{test}}})$ **then** ▷ *distribute reward*
    if $t < \mathsf{T}$ **then**
      $\mathbf{V}^{\mathsf{t}} \leftarrow \mathsf{QAudit}_{\mathfrak{v}}(\mathsf{R}, 0, \boldsymbol{\rho}^{\mathsf{t}}, \boldsymbol{\rho}^{\mathsf{t}+1}, \mathbf{B}^{\mathsf{t}}, \mathbf{B}^{\mathsf{t}+1})$
      $\mathbf{V} \leftarrow \left[..., \mathbf{V}^{\mathsf{t}}\right]$
    else
      $\mathbf{V}^{\mathsf{T}} \leftarrow \mathsf{QAudit}_{\mathfrak{v}}(\mathsf{R}, \mathfrak{v}_{\mathsf{pa}}, \boldsymbol{\rho}^{\mathsf{t}}, (), \mathbf{B}^{\mathsf{t}}, \{\})$
      $\mathbf{V} \leftarrow \left[..., \mathbf{V}^{\mathsf{T}}\right]$
    end if
    if $\varrho^{\mathsf{t}} > 0$ **then**
      $\left(\varrho^{\mathsf{t}}, \Delta^{\mathsf{t}}_{\mathsf{Agent}}, \Delta^{\mathsf{t}}_{\mathsf{Auditor}}\right) \leftarrow \mathsf{QAudit}_{\varrho\,\mathfrak{v}}\left(\mathsf{a} \mid\mid \mathsf{b}, \mathsf{R}, \mathbf{W}^{\mathsf{t}}, \boldsymbol{\rho}^{\mathsf{t}}, \mathbf{B}^{\mathsf{t}}, \mathbb{D}_{\mathsf{test}}, \mathbb{D}_{\mathsf{val}},\right)$
      $\left(\Delta, \Delta_{\mathsf{Agent}}, \Delta_{\mathsf{Auditor}}\right) \leftarrow \left([...\varrho^{\mathsf{t}}], \{..., \Delta^{\mathsf{t}}_{\mathsf{Agent}}\}, \{..., \Delta^{\mathsf{t}}_{\mathsf{Agent}}\}\right)$
    end if
  else ▷ *allocate regret penalty*
    if $\varrho^{\mathsf{t}} > 0$ **then**
      $\left(\varrho^{\mathsf{t}}, \Delta^{\mathsf{t}}_{\mathsf{Agent}}, \Delta^{\mathsf{t}}_{\mathsf{Auditor}}\right) \leftarrow \mathsf{QAudit}_{\varrho\,\mathfrak{v}}\left(\mathsf{a} \mid\mid \mathsf{b}, \mathsf{R}, \mathbf{W}^{\mathsf{t}}, \boldsymbol{\rho}^{\mathsf{t}}, \mathbf{B}^{\mathsf{t}}, \mathbb{D}_{\mathsf{test}}, \mathbb{D}_{\mathsf{val}},\right)$
      $\left(\Delta, \Delta_{\mathsf{Agent}}, \Delta_{\mathsf{Auditor}}\right) \leftarrow \left([...\varrho^{\mathsf{t}}], \{..., \Delta^{\mathsf{t}}_{\mathsf{Agent}}\}, \{..., \Delta^{\mathsf{t}}_{\mathsf{Agent}}\}\right)$
    end if
  end if
end for
return $(\Theta, \mathbf{W}, \mathbf{V}, \Delta, \Delta_{\mathsf{Agent}}, \Delta_{\mathsf{Auditor}})$

Between stages 1 and 2, QAudit samples from the generative models $\boldsymbol{\theta}$ and updates the discriminative model $\mathbf{w}$. During stage 2, the market reveals stage 1's submissions and accept wagers for top model, *while* accepting submissions in private for stage 2. This continues until stage/day 61, the market opens one last time to elicit wagers on models submitted during stage 60.

3. *Winners per stage* $\mathsf{R} \leq \frac{\mathsf{K}}{2}$. This constant determines the top R teams from each time stage that qualify for wagering rewards. Models submitted by the qualifying R Agents will be used to determine test error during the QAudit reward phase. QAudit competition is effectively memory-less:

   a) R is constant throughout the stages, so the Agents and their models are *not* eliminated across stages. It determines the top R experts (Agent-submitted models) from each stage, whose performance will be *collectively* evaluated on the test $\mathbb{D}_{\mathsf{test}}$ during the reward phase.

   b) Agents who were selected in the previous stage in the winning pool by Auditors are no more likely to be selected in the next stage. That is assuming the Auditors also wager in a manner that is conditionally independent across stages.

4. *Data set* $\mathbb{D}$ used to judge all the submitted models, and optional query set $\mathbb{Q}$. The two canonical QAudit use cases are:

   a) (*determine the quality of generative model*) of form $\mathfrak{p}(\phi_{\mathbf{x}_\tau}|\phi_{\mathbf{x}_{\tau-1}};\boldsymbol{\theta})$. That is to say the model accepts query $\mathbf{x}_{\tau-1} \in \mathbb{Q}$, and outputs a response $\mathbf{x}_\tau$. If the model is an "inverted-auto-captioning" system, then this query $\mathbf{x}_{\tau-1}$ may be: "generate an image of a horse riding a motorcycle on the beach," and the model will output real valued image $\mathbf{x}_\tau$ of just such an image [4]. The data set is then of form: $\mathbb{D} = \{(\mathbf{x}_{\tau-1}, \mathbf{x}'_\tau)_{\mathsf{j}}\}_{\mathsf{j}\geq 1}$, where $\mathbf{x}_{\tau-1}$ is human generated query prompt, and $\mathbf{x}'_\tau$ a *human* response. The model $\boldsymbol{\theta}$ is judged against this human response.

   b) (*buy data sets* $\mathbb{D}$). In this case the query set $\mathbb{Q}$ is empty, and the PA provides an example data set $\mathbb{D}$. This $\mathbb{D}$ is too small to be useful, so Agents are expected to expand its size and scope.

5. *Noise generator* $\boldsymbol{\eta}$ used to generate negative examples for consumption by $\mathsf{QAudit}_{\mathsf{train}}$ and $\mathsf{QAudit}_{\mathsf{infer}}$. They train a discriminative surrogate to determine the fidelity or quality of data sampled from Agent-submitted data or generative models. This setup superficially resembles generative adversarial networks (GANs) of [GP+14], but is in fact closer to online kernel regression as seen in [ZJ+12]. In this case data sampled from generative model $\boldsymbol{\theta}^{\mathsf{t}}_{\mathsf{k}}$ are used to train a discriminative model $\mathfrak{p}(\mathsf{y}|\phi_{\mathbf{x}};\mathbf{w}^{\mathsf{t}}_{\mathsf{k}})$. This discriminative surrogate $\mathbf{w}^{\mathsf{t}}_{\mathsf{k}}$ is then used as a proxy to judge quality of generative model $\boldsymbol{\theta}^{\mathsf{t}}_{\mathsf{k}}$ or data submitted by $\mathsf{Agent}_{\mathsf{k}}$ at time t. Note this noise generator $\boldsymbol{\eta}$ may simply generate random signal (i.e. "snow" in the case of images), or it may be an *earlier* version of some generative model, whose output are of "lower quality."

6. *Kernel embedding function* ϕ. Raw data $\mathbf{x}$ cannot train a surrogate discriminative classifier in a meaningful manner. Instead this $\mathbf{x}$ must be kernelized into $\phi_{\mathbf{x}}$. A data set $\mathbb{D}$ so kernelized will span a common space, and all surrogate model comparisons will occur in this space constructed by the data set [5] Note there is a subtlety here in that the Agent-submitted models $\boldsymbol{\theta}_{\mathsf{k}}$ are neural networks (ANN) that typically construct ϕ end-to-end in some opaque manner. In fact, many intermediate layers of said $\boldsymbol{\theta}_{\mathsf{k}}$ can be used as kernel mappings, see Appendix E.4 [[JG+18], [Yan20],[YL20],[AW+21],[HB+20],[AD+19], [Dom20]]. However we do not use the intermediate layers of the Agent-submitted models as feature mappings for a few reason:

[4] Note $\mathbf{x}_{\tau-1}$ is a real valued sentence vector, while $\mathbf{x}_\tau$ is a real valued $m \times n$ pixel image.
[5] See Appendix E for definition of Hilbert space, and Representer theorem.

a) doing so would entail a literal "change of frame-of-reference" when judging model quality. Instead the machine learning community has many such embedding models ϕ for every conceivable class of data. For example, if the models $\boldsymbol{\theta}_{\mathsf{k}}$ are language generative models, then "Word2Vec" may be one possible embedding model [MC+13]. [6] In practice, Metheus will have pre-set QAudit classes with community-approved embedding functions, so the principle auditor do not have be consider this problem.

b) The ANN may not be publicly viewable to QAudit protocol. Alternatively if the ANN is public, selecting the "right" layer may be labor intensive.

c) Agent may be submitting raw data instead of ANN. In which case some common feature map ϕ is needed to send all submitted data to a common Kernel space.

Commitment Phase

This is the pre-competition phase where data assets are prepared and the principle auditor is charged the bounty. The PA provided dataset $\mathbb{D}$ is partitioned into private test and public validation sets using subroutine $\mathsf{QAudit}_{\mathbb{D}}$. The validation set is revealed and used to judge the improvement in surrogate discriminative model from $\mathbf{w}_{\mathsf{k}}^{\mathsf{t}-1}$ to $\mathbf{w}_{\mathsf{k}}^{\mathsf{t}}$ on a preliminary basis. The test set $\mathbb{D}_{\mathsf{test}}$ remains hidden, and is revealed in the reward phase as the parimutuel rewards are calculated. See $\mathsf{QAudit}_{\mathbb{D}}$ for dataset construction process.

Protocol 2 $\mathsf{QAudit}_{\mathbb{D}}$ data commitment subprotocol

Require: data set $\mathbb{D}$, noise generator $\boldsymbol{\eta}$, embedding ϕ

$(\mathbb{D}_{\mathsf{test}}, \mathbb{D}_{\mathsf{val}}) \leftarrow ([], [])$
$\mathsf{k} \leftarrow 0$
for $\mathbf{x}_{\mathsf{i}} \in \mathbb{D}$ **do**
  $\mathbf{z}_{\mathsf{i}} \leftarrow \boldsymbol{\eta}()$ ▷ *out of class sample*
  let $v = \{(\phi_{\mathbf{x}_{\mathsf{i}}}, 1), (\phi_{\mathbf{z}_{\mathsf{i}}}, 0)\}$
  if $\mathsf{k} \leq \frac{|\mathbb{D}|}{4}$ **then**
    $(\mathbb{D}_{\mathsf{test}}, \mathbb{D}_{\mathsf{val}}) \leftarrow ([..., v], \mathbb{D}_{\mathsf{val}})$ ▷ *fill tests set*
  else
    $(\mathbb{D}_{\mathsf{test}}, \mathbb{D}_{\mathsf{val}}) \leftarrow (\mathbb{D}_{\mathsf{test}}, [..., v])$ ▷ *fill validation set*
  end if
  $\mathsf{k}+ = 1$
end for
return $(\mathbb{D}_{\mathsf{test}}, \mathbb{D}_{\mathsf{val}})$

Public Audit Phase

This phase lasts $\mathsf{T}+1$ time stages, and it alternatively fills the queue Θ with model checkpoints submitted by Agents from the current stage t, while the Auditors audit the checkpoints from the last time stage $\mathsf{t}-1$ by wagering on their quality. The data updated at each stage include:

1. (Θ *model parameter history*): history of model submission over all T stages. Each t'th element of Θ is a set Θ^{t} of length K, featuring model submission from all K Agents. Agents can submit as often as they like while the private submission phase is open, but QAudit only stores the latest submission.

$$\Theta = [\Theta^1, \ldots, \Theta^{\mathsf{T}}] \quad \text{where } \Theta^{\mathsf{t}} = \{\boldsymbol{\theta}_1^{\mathsf{t}}, \ldots, \boldsymbol{\theta}_{\mathsf{K}}^{\mathsf{t}}\}.$$

[6] Note Word2Vec is over 10 years old. In some sense QAudit is akin to giving SATs to machine learning models in *Latin*. It is not the definitive judge of quality, but it is a *standardized one*.

2. ($\mathbf{W}$ *classifier parameter history*): history of surrogate models over all T stages. Each t'th element of $\mathbf{W}$ is a set $\mathbf{W}^{\mathsf{t}}$ of length K, featuring model surrogates trained on submissions from all K Agents.

$$\mathbf{W} = [\mathbf{W}^1, \ldots, \mathbf{W}^{\mathsf{T}}] \quad \text{where } \mathbf{W}^{\mathsf{t}} = \{\mathbf{w}_1^{\mathsf{t}}, \ldots, \mathbf{w}_{\mathsf{K}}^{\mathsf{t}}\}.$$

$\mathbf{w}_{\mathsf{k}}^{\mathsf{t}}$ is trained from samples drawn from $\boldsymbol{\theta}_{\mathsf{k}}^{\mathsf{t}}$ using $\mathsf{QAudit}_{\mathsf{train}}$. The interpretation of $\mathbf{w}$ is as follows:

   a) in the case where QAudit is eliciting generative model from $\mathsf{Agent}_{\mathsf{k}}$, then vector $\mathbf{w}_{\mathsf{k}}^{\mathsf{t}}$ parameterize a logistic regression model of form $\mathfrak{p}(\mathsf{y}|\mathbf{x}; \mathbf{w}_{\mathsf{k}}^{\mathsf{t}})$, with $\mathsf{y} \in \{0, 1\}$ and some real valued $\mathbf{x} \in \Omega$. The model classifies the likelihood that $\mathbf{x}$ was generated by the model $\boldsymbol{\theta}_{\mathsf{k}}^{\mathsf{t}}$ given parameter $\mathbf{w}_{\mathsf{k}}^{\mathsf{t}}$ (see example E.3.3).

   b) In the case where QAudit elicits raw data from $\mathsf{Agent}_{\mathsf{k}}$ from some PA-defined class, then the $\mathfrak{p}(\mathsf{y}|\mathbf{x}; \mathbf{w}_{\mathsf{k}}^{\mathsf{t}})$ describes the likelihood that the data belongs to this class. For example, if the PA wish to buy images of "socks on the floor" from Agents to train a robot to pick up socks, then $\mathfrak{p}(\mathsf{y}|\mathbf{x}; \mathbf{w}_{\mathsf{k}}^{\mathsf{t}})$ is a classifier for socks, trained on images of socks submitted by $\mathsf{Agent}_{\mathsf{k}}$.

3. ($\mathsf{logprob}$ *log likelihood history*): storing how well every proxy classifier $\mathbf{w}_{\mathsf{k}}^{\mathsf{t}}$ performed on the *validation set* $\mathbb{D}_{\mathsf{val}}$, measured by the log-likelihood of sampling this data from nature as defined by $\mathfrak{p}(\mathsf{y}|\phi_{\mathbf{x}}; \mathbf{w}_{\mathsf{k}}^{\mathsf{t}})$. This loss is public and should be used by Auditors to wager on the underlying model $\boldsymbol{\theta}_{\mathsf{k}}^{\mathsf{t}}$.

$$\mathsf{logprob} = \left\{\mathsf{logprob}^1, \ldots, \mathsf{logprob}^{\mathsf{T}}\right\} \quad \text{where } \mathsf{logprob}^{\mathsf{t}} = \left\{\mathsf{logprob}_1^{\mathsf{t}}, \ldots, \mathsf{logprob}_{\mathsf{K}}^{\mathsf{t}}\right\}$$

4. ($\mathbf{B}$ *share history*): Stores the bid history throughout stage t for every $\mathsf{Auditor}_{\mathsf{i}}$. During stage t, each $\mathsf{Auditor}_{\mathsf{i}}$ submits bid vectors $\mathbf{b}_{\mathsf{i}}^{\mathsf{t}} = \{\mathsf{b}_{1,i}^{\mathsf{t}}, ..., \mathsf{b}_{\mathsf{k},\mathsf{i}}^{\mathsf{t}}\}$, where $\mathsf{b}_{\mathsf{k},\mathsf{i}}^{\mathsf{t}} \geq 0$ is a scalar value denoting the number of shares $\mathsf{Auditor}_{\mathsf{i}}$ purchased for model k at time t:

$$\mathbf{B} = \left[..., \left[\mathbf{b}_1^{\mathsf{t}}, ..., \mathbf{b}_{\mathsf{i}}^{\mathsf{t}}, ..., \mathbf{b}_{\mathsf{N}}^{\mathsf{t}}\right]_{\mathsf{t}}, \ldots\right]_{\mathsf{t}=1..\mathsf{T}+1} \quad \text{where } \mathbf{b}_{\mathsf{i}}^{\mathsf{t}} = (\mathsf{b}_{1,\mathsf{i}}^{\mathsf{t}}, ..., \mathsf{b}_{\mathsf{K},\mathsf{i}}^{\mathsf{t}}).$$

   Some additional notes:

   a) within each stage, the auditors may bid as often as they like, thus each $\mathbf{b}_{\mathsf{i}}^{\mathsf{t}}$ may appear more than once.

   b) The share $\mathsf{b}_{\mathsf{k},\mathsf{i}}^{\mathsf{t}}$ has associated purchase price ρ_i^{t} denominated in $\mathfrak{miat}$, thus the Auditor's purchase value for this vector is $\langle \mathbf{b}_{\mathsf{i}}^{\mathsf{t}}, \boldsymbol{\rho}^{\mathsf{t}} \rangle$.

5. ($\mathbf{P}$ *price history*) stores the price of buying a share of each model $\boldsymbol{\theta}_{\mathsf{k}}^{\mathsf{t}}$ at market close for stage t:

$$\mathbf{P} = \left[\boldsymbol{\rho}^1, \ldots, \boldsymbol{\rho}^{\mathsf{T}+1}\right] \quad \text{where } \boldsymbol{\rho}^{\mathsf{t}} = \left(\rho_1^{\mathsf{t}}, \ldots, \rho_{\mathsf{K}}^{\mathsf{t}}\right).$$

   The vector $\boldsymbol{\rho}^{\mathsf{t}}$ is initialized by subroutine $\mathsf{QAudit}_{\rho^\circ}$, it computes the Bregman divergence from $\mathbf{w}_{\mathsf{k}}^{\mathsf{t}-1}$ to $\mathbf{w}_{\mathsf{k}}^{\mathsf{t}}$. This is the initial price or prior likelihood that the model $\mathbf{w}_{\mathsf{k}}^{\mathsf{t}}$ improves the previous model $\mathbf{w}_{\mathsf{k}}^{\mathsf{t}-1}$. As the Agents submit buy orders, the current market price is updated by the logarithmic market scoring rule in $\mathsf{QAudit}_{\mathfrak{v}}$ to reflect the current market consensus.

For every $(\mathsf{t}-1, \mathsf{t})$ time stages, the competition alternate between two phases:

1. (*model elicitation step).* At time $\mathsf{t}-1$, Agents submit their checkpoints into $\Theta^{\mathsf{t}-1}$. Agents may submit as often as like, and the order of submission does not matter, as QAudit will build the final $\Theta^{\mathsf{t}-1}$ from the latest submission from each agent k at time t. Before $\Theta^{\mathsf{t}-1}$ is published, QAudit_{Θ} computes the proxy classifiers $\mathbf{W}^{\mathsf{t}-1}$, and publishes it along with associated log-data-likelihood history $\mathsf{logprob}^{\mathsf{t}-1}$. The losses

will be revealed to Auditors at market open in t, and maybe used as one indicator of quality, alongside other metrics, i.e. Frechet inception distance score of the submitted generative models. Note the surrogate model weights $\mathbf{W}^{\mathsf{t}-1}$ are not published as this would skew incentives, in that Agents may train models to "beat" the surrogate classifiers.

Protocol 3 QAudit_Θ Agent model elicitation subprotocol

Require: validation set $\mathbb{D}_{\mathsf{val}}$, noise generator $\boldsymbol{\eta}$
Require: surrogate model weights from previous stage $\{\mathbf{w}_{\mathsf{k}}^{\mathsf{t}-1}\}_{\mathsf{k}=1..\mathsf{K}}$
Require: agent accounts $\{\mathsf{Agent}_{\mathsf{k}}\}_{\mathsf{k}=1..\mathsf{K}}$
1: $(\Theta^{\mathsf{t}}, \mathbf{W}^{\mathsf{t}}, \mathsf{logprob}^{\mathsf{t}}) \leftarrow (\{\}, \{\}, [])$
2: **while** submission open **do**
3: $\quad \boldsymbol{\theta}_{\mathsf{k}}^{\mathsf{t}} \leftarrow \mathsf{Agent}_{\mathsf{k}}$
4: $\quad (\mathbf{w}_{\mathsf{k}}^{\mathsf{t}}, \mathsf{logprob}_{\mathsf{k}}^{\mathsf{t}}) \leftarrow \mathsf{QAudit}_{\mathsf{train}}(\boldsymbol{\theta}_{\mathsf{k}}^{\mathsf{t}}, \boldsymbol{\eta}, \mathbf{w}_{\mathsf{k}}^{\mathsf{t}-1}, \mathbb{D}_{\mathsf{val}}, \phi, 10)$
5: $\quad (\Theta^{\mathsf{t}}, \mathbf{W}^{\mathsf{t}}, \mathsf{logprob}^{\mathsf{t}}) \leftarrow (\{.., \boldsymbol{\theta}_{\mathsf{k}}^{\mathsf{t}}\}, \{..., \mathbf{w}_{\mathsf{k}}^{\mathsf{t}}\}, \{..., \mathsf{logprob}_{\mathsf{k}}^{\mathsf{t}}\})$
6: **end while**
7: **return** $(\Theta^{\mathsf{t}}, \mathbf{W}^{\mathsf{t}}, \mathsf{logprob}^{\mathsf{t}})$

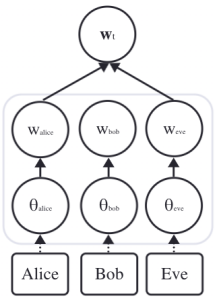


Figure 2.5: In every stage QAudit elicits generative models $\boldsymbol{\theta}$ from Agents such as alice, bob, and eve. Data is sampled from said generators to train three respective discriminative models $\mathfrak{p}(\mathsf{y}|\mathbf{x}; \mathbf{w}_{\mathsf{k}})$. In this diagram, the best $\mathsf{R} = 2$ performing discriminative models from alice and eve are used to build an ensemble model $\mathbf{w}^{\mathsf{t}}$ representing the winners of stage t. This ensemble is judged against the Auditor-selected ensemble in the previous stage $\mathbf{w}^{\mathsf{t}-1}$.

2. (*wager elicitation step).* At stage t, the previous submission stage's queue $\Theta^{\mathsf{t}-1}$ is published, along with losses $\mathsf{logprob}^{\mathsf{t}-1}$. $\mathsf{QAudit}_{\boldsymbol{\rho}^o}$ then initialize the price vector $\{\rho_{\mathsf{k}}^{\mathsf{t}-1}\}_{\mathsf{k}=1..\mathsf{K}}$ using the private parameter set $\mathbf{W}^{\mathsf{t}-1}$ and $\mathbf{W}^{\mathsf{t}-2}$. Note: QAudit is a memory-less protocol, so that the price vector is *re-initialized* on every stage. This means models who performed well with Auditors last stage are not at a price (dis)advantage this stage. The only factor (aside from the price ascending constant) that determine the initial price quote in every stage is the Bregman divergence between the model $\mathbf{w}_{\mathsf{k}}^{\mathsf{t}-1}$ in the current trading stage, and the corresponding model $\mathbf{w}_{\mathsf{k}}^{\mathsf{t}-2}$ from the previous trading stage (see Appendix (F.4)). Auditors approach the protocol and observes the model $\boldsymbol{\theta}_{\mathsf{k}}^{\mathsf{t}-1}$, in particular they see:

   a) current price $\rho_{\mathsf{k}}^{\mathsf{t}-1}$,
   b) log likelihood of data $\mathsf{logprob}_{\mathsf{k}}^{\mathsf{t}}$ computed on the validation set $\mathbb{D}_{\mathsf{val}}$,
   c) and any published metrics associated with model $\boldsymbol{\theta}_{\mathsf{k}}^{\mathsf{t}-1}$.

   Based on these metrics, Auditors may buy a share (denoted $\mathfrak{mint}_{\mathsf{k}}^{\mathsf{t}-1}$) of model $\boldsymbol{\theta}_{\mathsf{k}}^{\mathsf{t}-1}$ if they believe it is under-priced relative to its expected performance under auditor's belief. Each time the Auditor purchases shares of some model, its share price is updated by line 6-7 of $\mathsf{QAudit}_{\mathfrak{v}}$ using the logarithmic market scoring rule (LMSR). Each model in every submission stage are competing for R slots, and are eligible for parimutuel or mining rewards. Note the two steps are interleaved, so that i.e. if one time stage is 24 hours, then the Auditors are always trading yesterday's models.

Protocol 4 $\mathsf{QAudit}_{\mathfrak{v}}$ Auditor wager elicitation subprotocol with LMSR-based Amm

Require: dutch auction function v_{ρ}, price init constant v_{ρ}^{o}
Require: consecutive surrogate model weights $\mathbf{W}^{\mathsf{t}-1}$, and $\mathbf{W}^{\mathsf{t}}$
Require: log-data-likelihood logprob over models $\{\mathbf{w}_{\mathsf{k}}^{\mathsf{t}}\}_{\mathsf{k}=1..\mathsf{K}}$
Require: validation set $\mathbb{D}_{\mathsf{val}}$, auditor accounts $\{\mathsf{Auditor}_{\mathsf{i}}\}_{\mathsf{i}=1..\mathsf{N}}$
1: $\mathbf{B}^{\mathsf{t}} \leftarrow []$
2: $\left(\mathbf{s}^{\mathsf{t}},\ \boldsymbol{\rho}_{\tau_o}^{\mathsf{t}}\right) \leftarrow \mathsf{QAudit}_{\boldsymbol{\rho}^o}\left(\mathbb{D}_{\mathsf{val}}, \mathsf{v}_{\rho}^{o},\ \mathbf{W}^{\mathsf{t}-1}, \mathbf{W}^{\mathsf{t}}\right)$
3: **while** market open **do**
4: $\quad \mathbf{b}_{\mathsf{i}}^{\mathsf{t}} \leftarrow \mathsf{Auditor}_{\mathsf{i}}$ where $\mathbf{b}_{\mathsf{i}}^{\mathsf{t}} = (\mathsf{b}_{1,i}^{\mathsf{t}}, ..., \mathsf{b}_{\mathsf{k},\mathsf{i}}^{\mathsf{t}})$
5: $\quad$ let $\mathbf{s} = (.., \mathsf{s}_{\mathsf{k}}^{\mathsf{t}} + \mathsf{b}_{\mathsf{k},\mathsf{i}}^{\mathsf{t}}, ..)_{\mathsf{k}=1..\mathsf{K}}$
6: $\quad$ let $\mathsf{d} = \exp(\mathsf{s}_1^{\mathsf{t}}) + ... + \exp(\mathsf{s}_{\mathsf{k}}^{\mathsf{t}})$ where $\mathsf{s}_{\mathsf{k}}^{\mathsf{t}} \in \mathbf{s}$
7: $\quad$ let $\boldsymbol{\rho} = \mathsf{v}_{\rho}^{\mathsf{t}} \cdot \left(\frac{\exp(\mathsf{s}_1^{\mathsf{t}})}{\mathsf{d}}, \ldots, \frac{\exp(\mathsf{s}_{\mathsf{k}}^{\mathsf{t}})}{\mathsf{d}}\right)$ where $\mathsf{s}_{\mathsf{k}}^{\mathsf{t}} \in \mathbf{s}$ ▷ *update price by LMSR*
8: $\quad \left(\mathbf{s}^{\mathsf{t}},\ \boldsymbol{\rho}^{\mathsf{t}}\right) \leftarrow \left(\mathbf{s}, \boldsymbol{\rho}\right)$ ▷ *update current price for stage* t
9: $\quad \mathbf{B}^{\mathsf{t}} \leftarrow [..., \mathbf{b}_{\mathsf{i}}^{\mathsf{t}}]$
10: **end while**
11: **return** $\left(\mathbf{B}^{\mathsf{t}}, \boldsymbol{\rho}^{\mathsf{t}}\right)$

Commentary on the $\mathsf{QAudit}_{\mathfrak{v}}$ automated market maker (Amm):

a) $\mathsf{QAudit}_{\mathfrak{v}}$ references the data log-likelihood logprob computed by QAudit_{Θ} over the set of models $\left\{\mathbf{w}_{\mathsf{k}}^{\mathsf{t}}\right\}_{\mathsf{k}=1..\mathsf{K}}$, and suggests a set of possible rankings $\left\{\mathfrak{s}_{\mathsf{r}}^{*}\right\}_{\mathsf{r}\geq 1}$ over the K! possible permutation over Agents. This should aid the Auditors in deciding who to wager on.

b) The price vector $\boldsymbol{\rho}^{\mathsf{t}} = \left(\rho_1^{\mathsf{t}}, ..., \rho_{\mathsf{K}}^{\mathsf{t}}\right)$ is a probability distribution over the events:

$$\Omega = \left\{(\mathsf{Agent}_1, \mathsf{wins\ stage\ t}), ..., (\mathsf{Agent}_{\mathsf{K}}, \mathsf{wins\ stage\ t})\right\},$$

where every $\rho_{\mathsf{k}}^{\mathsf{t}} \in \boldsymbol{\rho}^{\mathsf{t}}$ denotes the likelihood that the kth Agent will submit the best model at stage t. Thus Auditors engaging in "subset betting," where the betting language is $\{(1,1)^{\mathsf{t}}, \ldots, (\mathsf{K},1)^{\mathsf{t}}\}$, each tuple k denotes $\mathsf{Agent}_{\mathsf{k}}$ will finish first at stage t. From the Bayesian perspective, the initial value of $\boldsymbol{\rho}^{\mathsf{t}}$ is the prior distribution, while every Auditor's buy signal is "data observation." Every buy vector $\mathbf{b}_{\mathsf{i}}^{\mathsf{t}} = (\mathsf{b}_{1,i}^{\mathsf{t}}, ..., \mathsf{b}_{\mathsf{K},i}^{\mathsf{t}})$ is filled with values $\mathsf{b}_{\mathsf{k},\mathsf{i}}^{\mathsf{t}} \geq 0$ denoting how many shares of $\mathsf{Agent}_{\mathsf{k}}$'s model the $\mathsf{Auditor}_{\mathsf{i}}$ would like to own. Recall the word "share" is synonymous with model token $\mathfrak{mint}_{\mathsf{k}}^{\mathsf{t}}$. [7]

c) Once the order for some model k is placed, the Agent receives $\mathsf{b}_{\mathsf{k},\mathsf{i}}^{\mathsf{t}}$ shares of the Machine Intelligence Nonfungible They point to a specific model $\mathbf{w}_{\mathsf{k}}^{\mathsf{t}}$ at stage t, and is a claim on the upside if model k is one of R top-ranked models in t. The $\mathsf{QAudit}_{\mathfrak{v}}$ Amm then incorporates the change in share inventory, and updates the market-estimated likelihood that any $\mathsf{Agent}_{\mathsf{k}}$ will win stage t. This "posterior update after data fusion" is written:

$$\mathfrak{p}(\mathbf{s}^{\mathsf{t}}|\mathbf{b}_{\mathsf{i}}^{\mathsf{t}}; \boldsymbol{\rho}^{\mathsf{t}}) = \left(\frac{\exp(\mathsf{s}_1^{\mathsf{t}} + \mathsf{b}_{1,i}^{\mathsf{t}})}{\sum_{\mathsf{k}=1}^{\mathsf{K}} \exp(\mathsf{s}_{\mathsf{k},\mathsf{i}}^{\mathsf{t}} + \mathsf{b}_{\mathsf{k},\mathsf{i}}^{\mathsf{t}})}, \ldots, \frac{\exp(\mathsf{s}_{\mathsf{K}}^{\mathsf{t}} + \mathsf{b}_{\mathsf{K}}^{\mathsf{t}})}{\sum_{\mathsf{k}=1}^{\mathsf{K}} \exp(\mathsf{s}_{\mathsf{k}}^{\mathsf{t}} + \mathsf{b}_{\mathsf{k},\mathsf{i}}^{\mathsf{t}})}\right).$$

Line 6-7 in $\mathsf{QAudit}_{\mathfrak{v}}$ computes this value up to some value $\mathsf{v}_{\rho}^{\mathsf{t}}$ that is a function of the current stage t. See example D.1.2.

[7] In some markets, auditors can sell shares back to the Amm, so that the Amm covers all buy/sell scenarios. $\mathsf{QAudit}_{\mathfrak{v}}$ does not offer this.

d) Now note the presence of PA-defined constant $\mathsf{v}_\rho^{\mathsf{t}}$ in the price update in line 7. In the canonical setting defined in example D.1.2, every value in the price vector is a true probability value. That is: $0 \leq \rho_{\mathsf{k}}^{\mathsf{t}} \leq 1$ for every $\rho_{\mathsf{k}}^{\mathsf{t}} \in \boldsymbol{\rho}^{\mathsf{t}}$, and the payout if event k occurs is \$1.00. However in our case the price is denominated in $\mathfrak{miat}$, and can be scaled by scalar value $\mathsf{v}_\rho^{\mathsf{t}}$ due to Dutch auction constraint. Similarly the reward is some variable amount of $\mathfrak{miat}$. Note $\mathsf{v}_\rho^{\mathsf{t}}$ is a function whose value differs as a function of stage t. Share prices increase monotonically in the increasing $\mathsf{v}_\rho^{\mathsf{t}}$ regime, so the cost of buying into each stage increases. Thus:

   i. incentivizing the auditors to participate in the earlier stages, as the cost of acquiring winning wagers is lower.

   ii. Allow auditors in every stage to profit as the overall betting pool increases each stage - assuming the betting volume is constant throughout the stages.

e) $\mathsf{QAudit}_{\mathfrak{v}}$ uses this descending likelihood estimation of every Agent's odd of winning the stage as a *heuristic* to order and select the top R teams. Notice this winning track wager is a far simpler betting language than permutation betting or subset betting [For+07]. For example if $\mathsf{QAudit}_{\mathfrak{v}}$ offered permutation betting instead, then it must offer $\mathsf{R}!$ options to express the likelihood of every ordering of the top R candidates. Depending on the user-defined R value, the PA may have to market an insurmountable array of Arrow-Debreu options.

f) Finally, observe the model $\boldsymbol{\theta}_{\mathsf{k}}^{\mathsf{t}}$ and $\mathbf{w}_{\mathsf{k}}^{\mathsf{t}}$ do not change within each wagering stage t, but its price does. $\mathsf{QAudit}_{\mathfrak{v}}$ Amm increases the price of the kth model each time Auditors buy shares of model k. The rational Auditor should buy early *within* each stage t at a lower price, if he believes model k is more performant than its price suggests. Recall QAudit is sequential parimutuel in payout for auditors, that is auditors are paid with a pro-rata percentage of the wagers on stage $\mathsf{t}+1$ in proportion to how many shares of the winning models he purchased on stage t. Therefore the auditor is rewarded for buying the right model early *within each stage* t. If he buys the top model $\mathbf{w}_{\mathsf{k}}^{\mathsf{t}}$ after other auditors have already moved, then the price of shares $\mathfrak{mint}_{\mathsf{k}}^{\mathsf{t}}$ would have increased, and his cost of buying the next stage's wager is higher. Conversely, when Auditors buy shares of model k, every model not indexed k decrease in price as $\boldsymbol{\rho}^{\mathsf{t}}$ is a scaled probability distribution. So a Auditor seeking model k may also wait until other Auditors have purchased other models.

Protocol 5 $\mathsf{QAudit}_{\boldsymbol{\rho}^o}$ price initialization subprotocol using Bregman Divergence

Require: validation set $\mathbb{D}_{\mathsf{val}}$, price init constant v_ρ^o

Require: consecutive surrogate model weights $\{\mathbf{w}_{\mathsf{k}}^{\mathsf{t}}\}_{\mathsf{k}=1..\mathsf{K}}$, and $\{\mathbf{w}_{\mathsf{k}}^{\mathsf{t}-1}\}_{\mathsf{k}=1..\mathsf{K}}$

1: $\left(\mathbf{s}, \boldsymbol{\rho}_{\tau_o}^{\mathsf{t}}\right) \leftarrow \left((), ()\right)$

2: **for** $\mathsf{k} \in 1...\mathsf{K}$ **do**

3: let $\rho_{\mathsf{k}}^{\mathsf{t}} = \mathsf{v}_\rho^o \mathsf{D}(\mathbf{w}_{\mathsf{k}}^{\mathsf{t}-1} || \mathbf{w}_{\mathsf{k}}^{\mathsf{t}})$ where

4:

$$\mathsf{D}(\mathbf{w}_{\mathsf{k}}^{\mathsf{t}-1} || \mathbf{w}_{\mathsf{k}}^{\mathsf{t}}) = \sum_{(\phi_{\mathbf{x}}, y) \in \mathbb{D}_{\mathsf{val}}} \mathfrak{p}\left(\mathsf{y} | \phi_{\mathbf{x}}; \mathbf{w}_{\mathsf{k}}^{\mathsf{t}-1}\right) \cdot \log \frac{\mathfrak{p}(\mathsf{y} | \phi_{\mathbf{x}}; \mathbf{w}_{\mathsf{k}}^{\mathsf{t}-1})}{\mathfrak{p}(\mathsf{y} | \phi_{\mathbf{x}}; \mathbf{w}_{\mathsf{k}}^{\mathsf{t}})}$$

5: $\left(\mathbf{s}, \boldsymbol{\rho}_{\tau_o}^{\mathsf{t}}\right) \leftarrow \left((... \log(\rho_{\mathsf{k}}^{\mathsf{t}})),\ (...\rho_{\mathsf{k}}^{\mathsf{t}}) \right)$

6: **end for**

7: **return** $(\mathbf{s}, \boldsymbol{\rho}_{\tau_o}^{\mathsf{t}})$

Commentary on $\mathsf{QAudit}_{\boldsymbol{\rho}^o}$:

a) Normally, a LMSR is initialized with zero inventory across all options offered. However in this case QAudit_{ρ^o} initializes market inventory state $\mathbf{s}^{\mathsf{t}}$ with the KL divergence between model $\mathbf{w}_{\mathsf{k}}^{\mathsf{t}-1}$ and $\mathbf{w}_{\mathsf{k}}^{\mathsf{t}}$, for every $\mathsf{k} = 1...\mathsf{K}$ (again scaled by a factor v_{ρ}^{o}). So that for example if QAudit_{ρ^o} places an order $\mathsf{b}_{\mathsf{k},\mathsf{Amm}}^{o}$, then $\mathsf{b}_{\mathsf{k},\mathsf{Amm}}^{o}$ shares of $\mathfrak{mint}_{\mathsf{k}}^{\mathsf{t}}$ model shares are minted and allocated to Metheus. Thus Metheus is a shareholder of every model k, in every stage t, across every instance of QAudit.

b) The initialized price vector is the prior likelihood of Agent' odds of winning. The implication is that every model update is an improvement, and the more an Agent updates a model, the more likely this model will win the stage. Naive Auditors may examine the price only and buy the cheapest model at the beginning of each stage. If the Agents expect this behavior, they may be incentivized to update their model $\boldsymbol{\theta}_{\mathsf{k}}^{\mathsf{t}-1}$ as little as possible. This incentive must then be balanced by the fact that the Agents also wish to make the top R slots per stage, so as to qualify for parimutuel rewards. The balance of the two opposing incentives is tilted by the regularization parameter R, defined by the principle auditor.

c) Bregman divergence or "KL-divergence in the primal form" computes the distance between two distributions. In this case, line 4 of protocol QAudit_{ρ^o} estimates the distance between two distributions by logistic regression models $\mathfrak{p}(\mathsf{y}|\phi_{\mathbf{x}}; \mathbf{w}_{\mathsf{k}}^{\mathsf{t}-1})$ and $\mathfrak{p}(\mathsf{y}|\phi_{\mathbf{x}}; \mathbf{w}_{\mathsf{k}}^{\mathsf{t}})$, in the space spanned by the data set $\mathbb{D}_{\mathsf{val}}$ kernalized by feature map ϕ. The regression models are trained in $\mathsf{QAudit}_{\mathsf{train}}$ by sampling from the Agent-submitted models $\boldsymbol{\theta}_{\mathsf{k}}^{\mathsf{t}}$ and $\boldsymbol{\theta}_{\mathsf{k}}^{\mathsf{t}-1}$, and the KL-divergence is estimated w.r.t $\mathbb{D}_{\mathsf{val}}$. See Appendix (F.4) and/or [WJ08] for derivation of KL divergence.

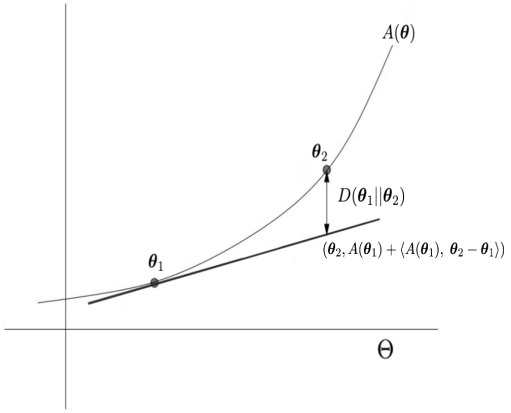


Figure 2.6: The Bregman divergence $\mathsf{D}(\boldsymbol{\theta}_1||\boldsymbol{\theta}_2)$ of two distributions parameterized by $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$, is the gap between the actual value of log-likelihood curve $\mathsf{A}(\boldsymbol{\theta}_2)$, and the first order polynomial approximation of value at $\mathsf{A}(\boldsymbol{\theta}_2)$, using the slope $\nabla\mathsf{A}(\boldsymbol{\theta}_1)$ at $\boldsymbol{\theta}_1$ [WJ04].

d) $\mathsf{QAudit}_{\mathfrak{v}}$ then initializes the market state vector:

$$\mathbf{s} = \left(\mathsf{s}_1^{\mathsf{t}}, ..., \mathsf{s}_{\mathsf{K}}^{\mathsf{t}}\right) \quad \text{where} \quad \mathsf{s}_{\mathsf{k}}^{\mathsf{t}} = \log\left(\mathsf{v}_{\rho}^{o}\mathsf{D}(\mathbf{w}_{\mathsf{k}}^{\mathsf{t}-1}||\mathbf{w}_{\mathsf{k}}^{\mathsf{t}})\right)$$

Note *the scaling factor* v_{ρ}^{o} *determines* Metheus*' margin per trade*, so that the larger the v_{ρ}^{o}, the greater the margin. This is because Auditors will be paid out of the wagering pool by the percentage of model shares they own, and QAudit is in effect "moving first" by purchasing $\log\left(\mathsf{v}_{\rho}^{o} \cdot \mathsf{D}(\mathbf{w}_{\mathsf{k}}^{\mathsf{t}-1}||\mathbf{w}_{\mathsf{k}}^{\mathsf{t}})\right)$ shares of model $\mathbf{w}_{\mathsf{k}}^{\mathsf{t}}$ at no cost.

e) In the naive implementation of exponential family prediction market of Appendix D, there are no Auditors. Instead every Agent submits his model $\mathbf{w}^{\mathsf{t}}$ with some stake that is computed with $\mathsf{D}(\mathbf{w}_{\mathsf{k}}^{\mathsf{t}-1}||\mathbf{w}_{\mathsf{k}}^{\mathsf{t}})$, this is a signal that the Agent has reported his true model. Translated into this context, when an Auditor buys the first share of model $\mathbf{w}_{\mathsf{k}}^{\mathsf{t}}$ at price $\log\left(\mathsf{v}_{\rho}^{1}\mathsf{D}(\mathbf{w}_{\mathsf{k}}^{\mathsf{t}-1}||\mathbf{w}_{\mathsf{k}}^{\mathsf{t}})\right)$, it is a signal that the Auditor believes the $\mathsf{Agent}_{\mathsf{k}}$ has revealed his best model at stage t.

Reward Allocation Phase

Protocol 6 $\mathsf{QAudit}_{\mathfrak{v}}$ Auditor/Agent reward allocation subprotocol

Require: regularization factor R, fiat denominated bounty ρ^{T}_{PA}
Require: final price vector at end of stage $(t, \mathsf{t}+1)$: $(\boldsymbol{\rho}^{\mathsf{t}}, \boldsymbol{\rho}^{\mathsf{t}+1})$
Require: agent accounts $\{\mathsf{Agent}_{\mathsf{k}}\}_{\mathsf{k}=1..\mathsf{K}}$, auditor accounts $\{\mathsf{Auditor}_{\mathsf{i}}\}_{\mathsf{i}=1..\mathsf{N}}$
Require: bid history at stage t and $\mathsf{t}+1$:

$$\mathbf{B}^{\mathsf{t}} = \{\mathbf{b}^{\mathsf{t}}_{\mathsf{i}}\}_{\mathsf{i}=1..\mathsf{N}} = \{\{\mathsf{b}^{\mathsf{t}}_{1,1}, .., \mathsf{b}^{\mathsf{t}}_{\mathsf{K},1}\}, .., \{\mathsf{b}^{\mathsf{t}}_{1,\mathsf{N}}, .., \mathsf{b}^{\mathsf{t}}_{\mathsf{K},\mathsf{N}}\}\}$$
$$\mathbf{B}^{\mathsf{t}+1} = \{\mathbf{b}^{\mathsf{t}}_{\mathsf{i}}\}_{\mathsf{i}=1..\mathsf{N}} = \{\{\mathsf{b}^{\mathsf{t}+1}_{1,1}, .., \mathsf{b}^{\mathsf{t}+1}_{\mathsf{K},1}\}, ..., \{\mathsf{b}^{\mathsf{t}+1}_{1,\mathsf{N}}, .., \mathsf{b}^{\mathsf{t}+1}_{\mathsf{K},\mathsf{N}}\}\}$$

$\{\mathsf{Agent}^{\mathsf{t}}_{\mathsf{r}}\}_{\mathsf{r}=1..\mathsf{R}} \leftarrow \mathbf{sort}\big(\{\mathsf{Agent}_{\mathsf{k}}\}_{\mathsf{k}=1..\mathsf{K}},\ \boldsymbol{\rho}^{\mathsf{t}}\big)[1:\mathsf{R}]$
$\{\mathsf{Agent}^{\mathsf{t}+1}_{s}\}_{\mathsf{s}=1..\mathsf{R}} \leftarrow \mathbf{sort}\big(\{\mathsf{Agent}_{\mathsf{k}}\}_{\mathsf{k}=1..\mathsf{K}},\ \boldsymbol{\rho}^{\mathsf{t}+1}\big)[1:\mathsf{R}]$
agentPairs $\leftarrow \mathbf{zip}\big(\{\mathsf{Agent}^{\mathsf{t}}_{\mathsf{r}}\}_{\mathsf{r}=1..\mathsf{R}},\ \{\mathsf{Agent}^{\mathsf{t}+1}_{s}\}_{\mathsf{s}=1..\mathsf{R}}\big)$
$\mathbf{V}^{\mathsf{t}} \leftarrow \big[(0x1, 0)_1, \ldots, (0x1, 0)_{\mathsf{R}\times(\mathsf{R}+\mathsf{N})}\big]$ ▷ *payouts with* $0x1$ *placeholder addresses*
▷ *select losing wagers on all* $(\mathsf{K}-\mathsf{R}+1)$ *models at stage* t *from* Auditor*s for agent payout*
$\{d^{\mathsf{t}}_{\mathsf{R}+1,1}, ..., d^{\mathsf{t}}_{\mathsf{K},\mathsf{N}}\} \leftarrow \mathbf{select}(\mathbf{B}^{\mathsf{t}}, [\mathsf{R}+1:\mathsf{K}])$

▷ *loop over* R *winners in pairwise descending order: ie* $(r, s) = (1, 1)$ $(r, s) = (\mathsf{R}, \mathsf{R})$
for $\big(\mathsf{Agent}^{\mathsf{t}}_{\mathsf{r}}, \mathsf{Agent}^{\mathsf{t}+1}_{s}\big) \in$ agentPairs **do**

$\{\mathsf{b}^{\mathsf{t}}_{\mathsf{r},1}, ..., \mathsf{b}^{\mathsf{t}}_{\mathsf{r},\mathsf{M}_{\mathsf{t}}}\} \leftarrow \mathbf{select}(\mathbf{B}^{\mathsf{t}}, [\mathsf{r}])$ ▷ *get all* M_{t} *wagers on model* R *at stage* t
$\big(\{\mathsf{logprob}^{\mathsf{t}}_{\mathsf{r}}\}_{\mathsf{r}=1..\mathsf{R}}, ...\big) \leftarrow \mathsf{QAudit}_{*}(\mathsf{R}, \mathbb{D}_{\mathsf{test}}, \mathbf{W}^{\mathsf{t}})$

$$\text{let } \mathfrak{v}^{\mathsf{t}}_{\mathsf{r}} = \underbrace{\frac{e^{\mathsf{logprob}^{\mathsf{t}}_{\mathsf{r}}}}{e^{\mathsf{logprob}^{\mathsf{t}}_{1}} + \ldots + e^{\mathsf{logprob}^{\mathsf{t}}_{\mathsf{R}}}} \cdot \rho^{\mathsf{t}}_{\mathsf{r}} \cdot (d^{\mathsf{t}}_{\mathsf{R}+1,1} + \ldots + d^{\mathsf{t}}_{\mathsf{K},\mathsf{N}})}_{\text{"eat what you bartered with"}} \cdot \underbrace{\frac{\mathfrak{fiat}}{\rho^{\mathsf{t}}_{\mathsf{r}}} \cdot \frac{\mathfrak{Fat}}{\mathfrak{fiat}} \cdot \frac{\mathfrak{miat}}{\mathfrak{Fat}}}_{\text{conversion}}$$

$\mathsf{Agent}_{\mathsf{r}} \leftarrow \mathsf{Agent}_{\mathsf{r}} + \mathfrak{v}^{\mathsf{t}}_{\mathsf{r}}$ ▷ *payout agent* r
$\mathbf{V}^{\mathsf{t}} \leftarrow \big[..., \big(\mathsf{Agent}_{\mathsf{r}}, \mathfrak{v}^{\mathsf{t}}_{\mathsf{r}}\big)\big]$

for $\mathsf{i} \in 1...\mathsf{N}$ **do** ▷ *reward* Auditor*s*
let $\mathfrak{v}^{\mathsf{t}}_{\mathsf{r},\mathsf{i}} = 0$
let $b^{\mathsf{t}}_{\mathsf{r},\mathsf{i}} \in \big(\mathsf{b}^{\mathsf{t}}_{1,\mathsf{i}}, .., \mathsf{b}^{\mathsf{t}}_{\mathsf{K},\mathsf{i}}\big)_{\mathsf{i}}$ ▷ *select auditor i's wager on model* r
let $\mathsf{q}^{\mathsf{t}}_{\mathsf{r},\mathsf{i}} = \mathsf{b}^{\mathsf{t}}_{\mathsf{r},\mathsf{i}}/\mathsf{s}^{\mathsf{t}}_{\mathsf{r}}$ where $\mathsf{s}^{\mathsf{t}}_{\mathsf{r}} \in \mathbf{s}^{\mathsf{t}}$ ▷ *auditor*$_i$*'s wager as* % *of all wagers on* r
if $(\rho^{\mathsf{T}}_{PA} = 0)$ **then** ▷ *parimutuel reward*

$$\mathfrak{v}^{\mathsf{t}}_{\mathsf{r},\mathsf{i}} = \underbrace{\mathsf{q}^{\mathsf{t}}_{\mathsf{r},\mathsf{i}} \cdot \mathsf{s}^{\mathsf{t}+1}_{\mathsf{s}} \cdot \rho^{\mathsf{t}+1}_{s}}_{\text{"eat what you paid for"}} \cdot \underbrace{\frac{\mathfrak{fiat}}{\rho^{\mathsf{t}+1}_{\mathsf{s}}} \cdot \frac{\mathfrak{Fat}}{\mathfrak{fiat}} \cdot \frac{\mathfrak{miat}}{\mathfrak{Fat}}}_{\text{conversion}} \quad \text{where } \mathsf{s}^{\mathsf{t}+1}_{\mathsf{s}} \in \mathbf{s}^{\mathsf{t}+1}$$

else ▷ *mining reward*

$$\mathfrak{v}^{\mathsf{t}}_{\mathsf{r},\mathsf{i}} = \underbrace{\mathsf{q}^{\mathsf{t}}_{\mathsf{r},\mathsf{i}} \cdot \mathfrak{v}_{\mathsf{pa}}}_{\%\text{ of bounty pool}} \cdot \underbrace{\frac{\mathfrak{Fat}}{\mathfrak{fiat}} \cdot \frac{\mathfrak{miat}}{\mathfrak{Fat}}}_{\text{conversion}}$$

end if
$\mathsf{Auditor}_{\mathsf{i}} \leftarrow \mathsf{Auditor}_{\mathsf{i}} + \mathfrak{v}^{\mathsf{t}}_{\mathsf{r},\mathsf{i}}$
$\mathbf{V}^{\mathsf{t}} \leftarrow \big[..., \big(\mathsf{Auditor}_{\mathsf{i}}, \mathfrak{v}^{\mathsf{t}}_{\mathsf{r},\mathsf{i}}\big)\big]$
end for
end for
return $\mathbf{V}^{\mathsf{t}}$

As the $\mathsf{T}+1$ elicitation stages end, QAudit replays the market history from $\mathsf{t} = \big[1, \ldots, \mathsf{T}+1\big]$, and distributes the rewards to all Auditors and Agents based on the quality of their wagers and submissions, respectively.

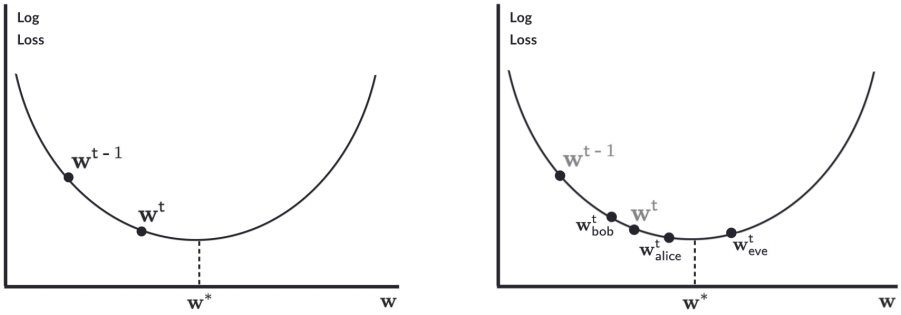


Figure 2.7: $\mathsf{QAudit}_{\mathsf{train}}$ trains logistic regression models of form $\mathfrak{p}(\mathsf{y}|\phi_{\mathbf{x}}; \mathbf{w}^{\mathsf{t}}_{\mathsf{k}}) = \boldsymbol{\sigma}(\langle \phi_{\mathbf{x}}, \mathbf{w}^{\mathsf{t}}_{\mathsf{k}} \rangle)$. This class of models have convex log-loss function due to the $\boldsymbol{\sigma}(\cdot)$ function. Over the $\mathsf{T}+1$ stages of QAudit, the protocol trains K different logistic regression models $\{\mathbf{w}_1, .., \mathbf{w}_{\mathsf{K}}\}$. Every model has a history of $(\mathsf{T}+1)$ iterations: $\{\mathbf{w}^1_{\mathsf{k}}, .., \mathbf{w}^{\mathsf{T}+1}_{\mathsf{k}}\}$. These models can be compared on the same log-loss curve, where lower values are better. In the left diagram, the ensemble model $\mathbf{w}^{\mathsf{t}}$ outperforms that of the previous stage $\mathbf{w}^{\mathsf{t}-1}$. So Agents whose models are used to build the ensemble $\mathbf{w}^{\mathsf{t}}$ are eligible for parimutuel rewards, as are the Auditors who wagered on them. In the right graph PA defined $\mathsf{R} = 3$, so that ensemble $\mathbf{w}^{\mathsf{t}}$ is constructed from models submitted by alice, eve, and bob. Since alice's model has the lowest log-loss value, she takes the most reward from the pot at stage t, eve take the 2nd most, and bob takes the least reward. Similarly, Auditors who wager on alice are eligible for the most sequential parimutuel reward drawn from stage $\mathsf{t}+1$, and so on. Note this comparison across all $(\mathsf{T}+1) \times \mathsf{K}$ models cannot be made with the underlying Agent-submitted generative models, as they do not have convex loss functions. See Appendix E.3.3 for more on kernel logistic regression, see D.2.3 for more on logistic regression and LMSR.

Protocol $\mathsf{QAudit}_{\mathfrak{v}}$ replays the wager history three stages at a time: $[\mathsf{t}-1, \mathsf{t}, \mathsf{t}+1]$, and compares the current surrogate composite model $\mathbf{w}^{\mathsf{t}}$ against the previous composite model $\mathbf{w}^{\mathsf{t}-1}$. This composite/ensemble model $\mathbf{w}^{\mathsf{t}}$ is constructed from the top R models $\{\mathbf{w}^{\mathsf{t}}_1, .., \mathbf{w}^{\mathsf{t}}_{\mathsf{R}}\}$. The prediction of the composite model on every point in the dataset $(\phi_{\mathbf{x}}, y) \in \mathbb{D}_{\mathsf{test}}$ is the weighted average of their individual predictions, weighed by the Auditors' wager on each model $\mathbf{w}^{\mathsf{t}}_{\mathsf{r}}$:

$$\mathfrak{p}\big(\mathsf{y}|\phi_{\mathbf{x}}; \mathbf{w}^{\mathsf{t}}\big) := \mathfrak{p}\big(\mathsf{y}|\phi_{\mathbf{x}}; \{\mathbf{w}^{\mathsf{t}}_{\mathsf{r}}\}_{\mathsf{r}=1..\mathsf{R}}\big) = \sum_{\mathsf{r}=1}^{\mathsf{R}} \frac{\rho^{\mathsf{t}}_{\mathsf{r}}}{\rho^{\mathsf{t}}_1 + .. + \rho^{\mathsf{t}}_{\mathsf{R}}} \cdot \mathfrak{p}(\mathsf{y}|\phi_{\mathbf{x}}; \mathbf{w}^{\mathsf{t}}_{\mathsf{r}}).$$

In other words, the more an Auditor wagers on the model $\boldsymbol{\theta}^{\mathsf{t}}_{\mathsf{r}}$, the more the prediction of surrogate classifier $\mathbf{w}^{\mathsf{t}}_{\mathsf{r}}$ affects the ensemble prediction.

Next, $\mathsf{QAudit}_{\mathfrak{v}}$ compares the data likelihood of test set $\mathbb{D}_{\mathsf{test}}$ for $\mathbf{w}^{\mathsf{t}-1}$ against $\mathbf{w}^{\mathsf{t}}$, and determine whether to reward the relevant parties. Under the principle of **"judge collectively but reward individually,"** if the current composite model $\mathbf{w}^{\mathsf{t}}$ is performant then:

1. every winning $\mathsf{Agent}_{\mathsf{r}}$ is paid from the losing wagers from the current pool at stage t, pro-rata to their respective model $\mathbf{w}^{\mathsf{t}}_{\mathsf{r}}$'s marginal information value add to the prediction of $\mathbf{w}^{\mathsf{t}}_{\mathsf{r}}$. So payout is parimutuel within every stage t for the winning Agents, and all payouts are denominated in $\mathfrak{miat}$. Note the $\mathsf{Agent}_{\mathsf{r}}$'s payout and his model prediction share is different:

$$\text{split to Auditor} = \frac{\rho^{\mathsf{t}}_{\mathsf{r}}}{\rho^{\mathsf{t}}_1 + ... + \rho^{\mathsf{t}}_{\mathsf{r}} + .. + \rho^{\mathsf{t}}_{\mathsf{R}}},$$

$$\text{split to Agent} = \frac{e^{\text{logprob}_{\mathsf{r}}^{\mathsf{t}}}}{e^{\text{logprob}_{1}^{\mathsf{t}}} + .. + e^{\text{logprob}_{\mathsf{r}}^{\mathsf{t}}} + .. + e^{\text{logprob}_{\mathsf{R}}^{\mathsf{t}}}}.$$

Moreover they are taken from different pools, the winning pool in the next stage for the Auditors and the losing pool from this stage for the Agents. Naturally, this discrepancy creates a misalignment of incentives between the Auditor and Agents. Thus regret penalty $\varrho_{\mathsf{r}}^{\mathsf{t}}$ is taken from the Auditor and given onto Agents to force an alignment.

Example 2.3.1. (Agent *submission scenario*) $\text{Agent}_{\mathsf{r}}$ submits model $\boldsymbol{\theta}_{\mathsf{r}}^{\mathsf{t}}$ at stage t, which is used to train the surrogate classifier $\mathbf{w}_{\mathsf{r}}^{\mathsf{t}}$. Now suppose $\mathbf{w}_{\mathsf{r}}^{\mathsf{t}}$ finished 2nd in the winning pool. Moreover, suppose the average $\mathfrak{miat}$ to $\mathfrak{Fat}$ conversion rate at end of last trading stage $\mathsf{T}+1$ is $110:1$. Then $\text{Agent}_{\mathsf{r}}$' payout is:

a) at market close for stage t, the total wager pool for model $\mathbf{w}_{\mathsf{r}}^{\mathsf{t}}$ has 250 shares, with an average price per share of 4.5 $\mathfrak{miat}$/share, for a total of

$$250\ \mathfrak{mint}_{\mathsf{r}}^{\mathsf{t}} \cdot \frac{4.5\mathfrak{miat}}{\mathfrak{mint}_{\mathsf{r}}^{\mathsf{t}}} = 1125\ \mathfrak{miat}$$

tokens in the pool.

b) Now suppose the principle auditor defined $\mathsf{R}=3$, and the total amount of $\mathfrak{miat}$ wagered over all three winning models at the current stage t is $3000\mathfrak{miat}$, including model-R. Next, suppose the remaining losing Agents' models have collectively elicited 5000 $\mathfrak{miat}$ in wagers at time t. Then $\text{Agent}_{\mathsf{r}}$' pro-rata share of the losing pool is:

$$\frac{1125}{3000} \cdot 5000\mathfrak{miat} = 1875\mathfrak{miat}.$$

This converts to fiat at:

$$1875\ \mathfrak{miat} \cdot \frac{1\mathfrak{Fat}}{110\mathfrak{miat}} \cdot \frac{1\mathsf{fiat}}{1\mathfrak{Fat}} = \mathsf{fiat}17.046.$$

Note although $\text{Agent}_{\mathsf{r}}$' pro-rata share of the losing pool is determined by how much the Auditors' wagered on his model, the agent himself have not wagered any amount of money, be it in fiat or $\mathfrak{miat}$. He only contributed time and effort, and yet have pocketed 17.05 in fiat in this scenario. In real life, this amount can be credited to $\text{Agent}_{\mathsf{r}}$' developer account, so that his future training sessions are subsided. This is an important point: **trading subsidize training**.

2. The winning Auditors who wagered on model $\mathbf{w}_{\mathsf{r}}^{\mathsf{t}}$ are paid from the *corresponding* betting pool of model $\mathbf{w}_{s}^{\mathsf{t}+1}$, where r and s are the matching positional index of the model rankings. So that QAudit is sequential parimutuel for the winning Auditors, and payouts are denominated in $\mathfrak{miat}$.

Example 2.3.2. (Auditor *wager scenario*) Suppose $\text{Auditor}_{\mathsf{i}}$ owns shares in model $\mathbf{w}_{\mathsf{r}}^{\mathsf{t}}$. As before, assume $\mathbf{w}_{\mathsf{r}}^{\mathsf{t}}$ finished 2nd in the winning pool:

a) $\text{Auditor}_{\mathsf{i}}$ owns 100 shares of $\mathbf{w}_{\mathsf{r}}^{\mathsf{t}}$, which he purchased at 3 $\mathfrak{miat}$ per share. So his total spend is:

$$100\ \mathfrak{mint}_{\mathsf{r}}^{\mathsf{t}} \cdot \frac{3\mathfrak{miat}}{1\mathfrak{mint}_{\mathsf{r}}^{\mathsf{t}}} = 300\ \mathfrak{miat}.$$

b) $\text{Auditor}_{\mathsf{i}}$ purchased the $\mathfrak{miat}$ at 100:1 $\mathfrak{miat}$ to $\mathfrak{Fat}$ conversion rate, and $\mathfrak{Fat}:\mathfrak{miat}$ conversion rate is pegged at 1:1 as $\mathfrak{Fat}$ is a stable coin. The auditor's fiat spend was:

$$300\mathfrak{miat} \cdot \frac{1\mathfrak{Fat}}{100\mathfrak{miat}} \cdot \frac{1\mathsf{fiat}}{1\mathfrak{Fat}} = \mathsf{fiat}3.00$$

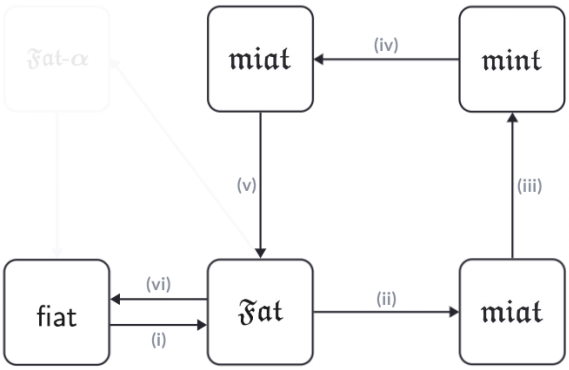


Figure 2.8: The QAudit conversion cycle:

i. QAudit wager conversion process begins when Auditors buy into QAudit with fiat, which converts to "phantom $\mathfrak{Fat}$" token.

ii. Conceptually, this phantom $\mathfrak{Fat}$ is minted and exchanged for $\mathfrak{miat}$ at the current floating exchange rate. This $\mathfrak{Fat}$ is then immediately burnt so that total $\mathfrak{Fat}$ supply stays constant.

iii. Auditors use $\mathfrak{miat}$ to wager on models by buying machine intelligence nonfungible tokens, or $\mathfrak{mint}$s from Agents. If the respective models perform, then Auditors win more $\mathfrak{miat}$.

iv. Auditors receive their reward by depositing their $\mathfrak{mint}$ and receive $\mathfrak{miat}$ owed to them. New $\mathfrak{miat}$ is minted and issued to winners. This is called "$\mathfrak{miat}$-mining." Observe as models improve, fresh $\mathfrak{miat}$ is mined.

v. Auditors may convert their $\mathfrak{miat}$ back to $\mathfrak{Fat}$ at the prevailing conversion rate. The $\mathfrak{miat}$ is recycled back to the global pool awaiting resale.

vi. Finally, this $\mathfrak{Fat}$ is converted back to pegged-fiat at 1:1 exchange rate.

Auditors whose winnings have been diluted away by $\mathfrak{miat}$ inflation may recover this loss by purchasing $\mathfrak{Fat}$-$\boldsymbol{\alpha}$ tokens and receiving inflation dividends. This is depicted in the faded bubble. Finally, there is an anti-deflation clause whereby if the $\mathfrak{miat}$:$\mathfrak{Fat}$ exchange rate falls within the conversion cycle duration, then Auditors/Agents must liquidate their position at an exchange rate no lesser than what they bought into the market. See eqn 4.2 and fig 4.6 in chapter 4 for more information.

c) At market close of stage t, the price $\rho_s^{\mathsf{t}} \in \boldsymbol{\rho}^{\mathsf{t}}$ of $\mathbf{w}_{\mathsf{r}}^{\mathsf{t}}$ is 5 $\mathfrak{miat}$ per share. This implies $\mathsf{Auditor}_{\mathsf{i}}$ purchased his shares at a lower price. The total wager pool for model $\mathbf{w}_{\mathsf{r}}^{\mathsf{t}}$ has 250 shares at market close for time t, with an average price per share of 4.5 $\mathfrak{miat}$/share, for a total of

$$250\ \mathfrak{mint}_{\mathsf{r}}^{\mathsf{t}} \cdot \frac{4.5\mathfrak{miat}}{1\mathfrak{mint}_{\mathsf{r}}^{\mathsf{t}}} = 1125\ \mathfrak{miat}$$

tokens in the pool. Moreover, suppose the average $\mathfrak{miat}$ to $\mathfrak{Fat}$ conversion rate for the pool is $105:1$, this could occur if $\mathfrak{miat}$ is mined over the wagering stage. Thus the fiat value of the pool is:

$$1125\ \mathfrak{miat} \cdot \frac{1\mathfrak{Fat}}{105\mathfrak{miat}} \cdot \frac{1\mathsf{fiat}}{1\mathfrak{Fat}} = \mathsf{fiat}10.71.$$

d) Next, the corresponding 2nd place finish model $\mathbf{w}_s^{t+1}$ at time stage $\mathsf{t}+1$ features a total wagering pool of 300 shares of $\mathfrak{mint}_{\mathsf{s}}^{\mathsf{t}+1}$, with average price/share of 6 $\mathfrak{miat}$, so that its $\mathfrak{miat}$ pool value is:

$$300\ \mathfrak{mint}_{\mathsf{r}}^{\mathsf{t}} \cdot \frac{6\mathfrak{miat}}{1\mathfrak{mint}_{\mathsf{s}}^{\mathsf{t}+1}} = 1800\ \mathfrak{miat}.$$

The $\mathfrak{miat}$ to $\mathfrak{Fat}$ average conversion at stage $\mathsf{t}+1$ is $110:1$, so the fiat value of the pool is:

$$1800\ \mathfrak{miat} \cdot \frac{1\mathfrak{Fat}}{110\mathfrak{miat}} \cdot \frac{1\mathsf{fiat}}{1\mathfrak{Fat}} = \mathsf{fiat}16.36.$$

e) Now suppose the collective model $\mathbf{w}^{\mathsf{t}}$, of which $\mathbf{w}_{\mathsf{r}}^{\mathsf{t}}$ is a component outperformed the respective model $\mathbf{w}^{\mathsf{t}-1}$. Then at reward time $\mathsf{Auditor}_{\mathsf{i}}$'s payout for model $\mathbf{w}_{\mathsf{r}}^{\mathsf{t}}$ is:

$$\mathfrak{v}_{\mathsf{s,i}}^{\mathsf{t}} = \frac{300\mathfrak{miat}}{1125\mathfrak{miat}} \cdot 1800\mathfrak{miat} = 480\mathfrak{miat}.$$

Observe $\mathsf{Auditor}_{\mathsf{i}}$ is only raking a percentage of the corresponding sth model at $\mathsf{t}+1$, namely $\mathbf{w}_s^{\mathsf{t}+1}$. $\mathsf{Auditor}_{\mathsf{i}}$ is *not* entitled to what was bet on the 1st, 3rd,..,Rth winning model at stage $\mathsf{t}+1$. This is "judge collectively but reward individually." Now his reward may be converted back to fiat, as more $\mathfrak{miat}$ mining reward has been minted, the $\mathfrak{miat}$ to $\mathfrak{Fat}$ conversion is now $112:1$. So $\mathsf{Auditor}_{\mathsf{i}}$'s intake is:

$$480\mathfrak{miat} \cdot \frac{1\mathfrak{Fat}}{112\mathfrak{miat}} \cdot \frac{1\mathsf{fiat}}{1\mathfrak{Fat}} = \mathsf{fiat}4.285.$$

So his profit is $4.285 - 3.00 = \mathsf{fiat}1.285$. Since the Auditors hold $\mathfrak{mint}_{\mathsf{k}}^{\mathsf{t}}$ shares, the Auditors must deposit $\mathfrak{mint}_{\mathsf{k}}^{\mathsf{t}}$ tokens to receive $\mathfrak{miat}$, which then converts to $\mathfrak{Fat}$ and fiat. Once deposited, QAudit burns the $\mathfrak{mint}_{\mathsf{k}}^{\mathsf{t}}$ shares. If however model k does not achieve one of the top R spots at time t, then the $\mathfrak{mint}_{\mathsf{k}}^{\mathsf{t}}$ shares all have value zero. The $\mathfrak{mint}$ are options on the upside.

3. In the terminal stage T, the winning Agents are paid out of the current stage as before, but the Auditors are paid out of the bounty $\mathfrak{v}_{\mathsf{pa}}$ paid by the principle auditor. As before, this bounty is converted to $\mathfrak{miat}$ and paid to Auditors, that is $\mathfrak{miat}$ is mined. This increases the total $\mathfrak{miat}$ supply. See $\mathfrak{v}_{\mathsf{r,i}}^{\mathsf{t}}$ of line 23.

4. Agents who submit losing models are not rewarded, however they are not eliminated either. Meanwhile Auditors who bet on losing models lose their wagers.

In QAudit, Auditors profit when:

1. the model they wagered on is one of the top R winning models at stage t.

2. The corresponding winning model at the same position at stage $\mathsf{t}+1$ has a bigger pool in proportion to what Auditors contributed on this current pool.

3. They buy shares at a lower price than its final price at market close for time t.

Conversely, adverse events that affect auditor's profit (aside from losing) are:

1. (*dilution*). $\mathfrak{miat}$ is mined or minted throughout the competition. And there may be many competitions occurring concurrently, each one of which is mining. Thus growth in $\mathfrak{miat}$ supply relative to $\mathfrak{Fat}$ supply may dilute the auditor's winning.

2. (*contracting wagering pool size across stages*). QAudit is sequential parimutuel in payout, thus the auditor's earnings depend on the next stage's wagers. If the crowd lose interest and wagers less and less, then Auditors may lose money even if they select the proper winning model.

3. (*compression of share price across stages)*. If price per share of models reduces across stages, then Auditors may lose money even if the overall pool size measured in shares do not contract. This may be remedied by price floor function $\mathsf{v}_{\rho}^{\mathsf{t}}$, which could force the share price to increase across stages t. In the event where purchasing volume is not decreasing, $\mathsf{v}_{\rho}^{\mathsf{t}}$ turns QAudit into an ascending price auction, whereby on every stage the price of the underlying model $\boldsymbol{\theta}_{\mathsf{k}}^{\mathsf{t}}$ increases. Increase $\mathsf{v}_{\rho}^{\mathsf{t}}$ values forces the Auditors to buy shares of the model early across the stages.

Factors influencing Auditor/Agent's profits are also *margin opportunities* for Metheus:

1. ($\mathfrak{miat}$ *dilution*). As total $\mathfrak{miat}$ supply expands relative to $\mathfrak{Fat}$ supply, Auditors who buy in any any given conversion rate are diluted. So i.e. if he who buys in at $100:1$ $\mathfrak{miat}$ to $\mathfrak{Fat}$, and then cashing out at $120:1$ $\mathfrak{miat}$ to $\mathfrak{Fat}$ conversion, then this delta contributes to Metheus' margin.

2. (*price initialization constant* v_{ρ}^{o}). This constant is invoked on every tth stage for a total of T stages. In effect, Metheus is buying shares of every model at each tth stage with the buy vector $\mathbf{b}^{\mathsf{t}} = \left(\mathsf{b}_{,1}^{\mathsf{t}}, ..., \mathsf{b}_{,\mathsf{K}}^{\mathsf{t}}\right)$ for free. So that at the end of the competition, Metheus is holding a set of shares: $\{\mathbf{b}^{1}, ..., \mathbf{b}^{\mathsf{T}}\}$ over every model in every stage. These shares dilute every Auditor and Agent's claim on the overall wagering pool over the T stages.

3. (2.5% *trading fee*). This fee is charged when Auditors buy in with fiat.

Remark 2.3.3. (*Redemption of model nonfungible token* $\mathfrak{mint}$) One aspect is missing in the documentation thus far: how Auditors redeem their token $\mathfrak{mint}$ for $\mathfrak{miat}$ currency. Recall that $\mathfrak{mint}_{\mathsf{r}}^{\mathsf{t}}$ is a *conditional* claim on $\mathfrak{miat}$ in the event its associated model $\boldsymbol{\theta}_{\mathsf{r}}^{\mathsf{t}}$ wins in stage t. Suppose the model $\boldsymbol{\theta}_{\mathsf{r}}^{\mathsf{t}}$ does win, and there are in total 100 shares of $\mathfrak{mint}_{\mathsf{r}}^{\mathsf{t}}$ claiming 2000 $\mathfrak{miat}$ tokens. Then if $\mathsf{Auditor}_{\mathsf{i}}$ holds 10 shares, he is entitled to $10/100 \cdot 2000$ $\mathfrak{miat}$ tokens. Meanwhile Auditor holding losing shares $\mathfrak{mint}_{\mathsf{s}}^{\mathsf{t}}$ $(s \neq r)$ are entitled to nothing. Their fiat wagers denominated in $\mathfrak{miat}$ are instead given to winning Agents in stage t.

Regret Adjustment Phase

After every reward payment stage t, there is a regret adjustment phase, whereby Agents who should have been selected as one of the top winning R models are paid with regret penalty taken from the winning Auditors. The operative word here is *should have*:

1. Agents who were already chosen by Auditors are not compensated with regret.
2. If the individual models in the worse ensemble are ranked according to one of the suggested rankings, then the Auditors are not penalized.

Regret penalty is designed specifically to dis-incentivize malicious Auditors from pumping the market by purchasing cheap and low quality model shares, thus manipulating the rankings and causing the Agents who submit good models to exodus the market.

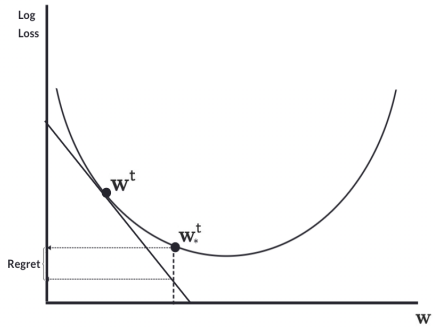


Figure 2.9: This diagram shows the difference in performance between the two models in the log-loss curve. However in QAudit_ϱ we compute the KL divergence $\mathsf{D}(\mathbf{w}^{\mathsf{t}}||\mathbf{w}_*^{\mathsf{t}})$. Remark E.3.4 states that they are comparable.

Protocol 7 QAudit_* maximum performing ensemble subroutine

Require: regularization parameter $\mathsf{R} \leq \mathsf{K}/2$
Require: model parameters $\{\mathbf{w}_{\mathsf{k}}^{\mathsf{t}}\}_{\mathsf{k}=1..\mathsf{K}}$, test set $\mathbb{D}_{\mathsf{test}}$
$\mathsf{logprob}s \leftarrow []$ ▷ *log-likelihood on test set for every model*
for $\mathbf{w}_{\mathsf{k}}^{\mathsf{t}} \in \{\mathbf{w}_{\mathsf{k}}^{\mathsf{t}}\}_{\mathsf{k}=1..\mathsf{K}}$ **do**
  let $\mathsf{logprob}_{\mathsf{k}}^{\mathsf{t}} = \sum_{(\phi_{\mathbf{x}},y)\in\mathbb{D}_{\mathsf{test}}} \log \mathfrak{p}(\mathsf{y}|\phi_{\mathbf{x}};\mathbf{w}_{\mathsf{k}}^{\mathsf{t}})$ where

$$\log \mathfrak{p}(\mathsf{y}|\phi_{\mathbf{x}};\mathbf{w}_{\mathsf{k}}^{\mathsf{t}}) = \mathsf{y}\log\boldsymbol{\sigma}(\langle\mathbf{w}_{\mathsf{k}}^{\mathsf{t}},\phi_{\mathbf{x}}\rangle) + (1-\mathsf{y})\log\boldsymbol{\sigma}(1-\langle\mathbf{w}_{\mathsf{k}}^{\mathsf{t}},\phi_{\mathbf{x}}\rangle)$$

  $\mathsf{logprob}s \leftarrow [..., \mathsf{logprob}_{\mathsf{k}}^{\mathsf{t}}]$
end for
$\{\mathbf{w}_{\mathsf{s}}^{\mathsf{t}}\}_{\mathsf{s}=1..\mathsf{R}} \leftarrow \mathbf{sort}\big(\{\mathbf{w}_{\mathsf{k}}^{\mathsf{t}}\}_{\mathsf{k}=1..\mathsf{K}},\ \mathsf{logprob}s\big)[1:\mathsf{R}]$ ▷ *select best* R *models in hindsight*
let $\mathsf{logprob}^{\mathsf{t}*} = \sum_{(\phi_{\mathbf{x}},y)\in\mathbb{D}_{\mathsf{test}}} \log \mathfrak{p}(\mathsf{y}|\phi_{\mathbf{x}};\mathbf{w}^{\mathsf{t}})$ where

$$\begin{aligned}\log \mathfrak{p}(\mathsf{y}|\phi_{\mathbf{x}};\mathbf{w}^{\mathsf{t}}) &= \mathsf{y}\log\mathfrak{p}(\mathsf{y}=1|\phi_{\mathbf{x}};\mathbf{w}^{\mathsf{t}}) + (1-\mathsf{y})\log\mathfrak{p}(\mathsf{y}=0|\phi_{\mathbf{x}};\mathbf{w}^{\mathsf{t}})\\ &= \mathsf{y}\log\left(\sum_{\mathsf{s}}^{\mathsf{R}}\beta_{\mathsf{s}}\boldsymbol{\sigma}(\langle\mathbf{w}_{\mathsf{s}}^{\mathsf{t}},\phi_{\mathbf{x}}\rangle)\right) + (1-\mathsf{y})\log\left(\sum_{\mathsf{s}}^{\mathsf{R}}\beta_{\mathsf{s}}\boldsymbol{\sigma}(1-\langle\mathbf{w}_{\mathsf{s}}^{\mathsf{t}},\phi_{\mathbf{x}}\rangle)\right)\end{aligned}$$

$$\text{and } \beta_{\mathsf{s}} = \frac{e^{\mathsf{logprob}_{\mathsf{s}}^{\mathsf{t}}}}{e^{\mathsf{logprob}_{1}^{\mathsf{t}}} + ... + e^{\mathsf{logprob}_{\mathsf{R}}^{\mathsf{t}}}}$$

return $\big(\mathsf{logprob}s, \mathsf{logprob}^{\mathsf{t}*}, \{\mathbf{w}_{\mathsf{s}}^{\mathsf{t}}\}_{\mathsf{s}=1..\mathsf{R}}\big)$

Commentary on QAudit_*: it selects best R classifier models based on their individual performance on the test set $\mathsf{logprob}_{\mathsf{s}}^{\mathsf{t}}$. QAudit_* then uses the weighted ensemble of $\mathbf{w}_{\mathsf{s}}^{\mathsf{t}}$'s to predict the label y given some input $\phi_{\mathbf{x}}$ drawn from the test set $\mathbb{D}_{\mathsf{test}}$. Notice the weight β_{s} is the individual model performance of each $\mathbf{w}_s^{\mathsf{t}}$.

Protocol 8 $\mathsf{QAudit}_{\varrho}$ compute Bregman's regret subprotocol

Require: test set $\mathbb{D}_{\mathsf{test}}$, validation set $\mathbb{D}_{\mathsf{val}}$, regularization parameter $\mathsf{R} \leq \mathsf{K}/2$,
Require: acceptable permutation difference $\mathsf{a} \,||\, \mathsf{b}$
Require: model params $\mathbf{W}^{\mathsf{t}}$, price vector for stage t at market close: $\boldsymbol{\rho}^{\mathsf{t}}$

1: $(\mathsf{logprob}^{\mathsf{t}}_{val}, \{\mathbf{w}^{\mathsf{t}}_u\}_{u=1..\mathsf{R}}) \leftarrow \mathsf{QAudit}_{\mathsf{infer}}\big(\mathsf{R}, \mathbb{D}_{\mathsf{val}}, \boldsymbol{\rho}^{\mathsf{t}}, \mathbf{W}^{\mathsf{t}}\big)$
2: $(\mathsf{logprob}^{\mathsf{t}}_{test}, \{\mathbf{w}^{\mathsf{t}}_{\mathsf{r}}\}_{\mathsf{r}=1..\mathsf{R}}) \leftarrow \mathsf{QAudit}_{\mathsf{infer}}\big(\mathsf{R}, \mathbb{D}_{\mathsf{test}}, \boldsymbol{\rho}^{\mathsf{t}}, \mathbf{W}^{\mathsf{t}}\big)$
3: $\big(\mathsf{logprob}s, \mathsf{logprob}^{\mathsf{t}}_{*}, \{\mathbf{w}^{\mathsf{t}}_{\mathsf{s}}\}_{\mathsf{s}=1..\mathsf{R}}\big) \leftarrow \mathsf{QAudit}_{*}(\mathsf{R}, \mathbb{D}_{\mathsf{test}}, \mathbf{W}^{\mathsf{t}})$

4: $\varrho^{\mathsf{t}} \leftarrow 0$ ▷ *total regret denominated in* $\mathfrak{miat}$
5: $\{\beta_{\mathsf{s}}\}_{\mathsf{s}=1..\mathsf{R}} \leftarrow [0]_{\mathsf{s}=1...\mathsf{R}}$
6: let $\mathfrak{s}_{val} = \mathsf{induceOrdering}\big(\{\mathbf{w}^{\mathsf{t}}_u\}_{u=1..\mathsf{R}}\big)$
7: let $\mathfrak{s}_{test} = \mathsf{induceOrdering}\big(\{\mathbf{w}^{\mathsf{t}}_{\mathsf{r}}\}_{\mathsf{r}=1..\mathsf{R}}\big)$
8: **if** $\exp(\mathsf{logprob}^{\mathsf{t}}_{*}) > \exp(\mathsf{logprob}^{\mathsf{t}}_{test})$ & $\big(\mathfrak{s}_{val} \,||\, \mathfrak{s}_{test}\big) > (\mathsf{a} \,||\, \mathsf{b})$ **then**
9: $\varrho^{\mathsf{t}} \leftarrow \mathsf{v}_{\rho}^{\mathsf{t}+1} \cdot \mathsf{D}(\mathbf{w}^{\mathsf{t}}||\mathbf{w}^{\mathsf{t}}_{*})$ where ▷ *regret is kl-div between models*

$$\mathsf{D}(\mathbf{w}^{\mathsf{t}}||\mathbf{w}^{\mathsf{t}}_{*}) = \sum_{(\phi_{\mathbf{x}},y)\in\mathbb{D}_{\mathsf{test}}} \mathfrak{p}\big(\mathsf{y}|\phi_{\mathbf{x}};\mathbf{w}^{\mathsf{t}}\big)\log\frac{\mathfrak{p}(\mathsf{y}|\phi_{\mathbf{x}};\mathbf{w}^{\mathsf{t}})}{\mathfrak{p}(\mathsf{y}|\phi_{\mathbf{x}};\mathbf{w}^{\mathsf{t}}_{*})} \quad \text{where}$$

$$\mathfrak{p}\big(\mathsf{y}|\phi_{\mathbf{x}};\mathbf{w}^{\mathsf{t}}\big) = \mathsf{y}\log\bigg(\sum_{\mathsf{r}}^{\mathsf{R}}\alpha_{\mathsf{r}}\boldsymbol{\sigma}(\langle\mathbf{w}^{\mathsf{t}}_{\mathsf{r}},\phi_{\mathbf{x}}\rangle)\bigg) + (1-\mathsf{y})\log\bigg(\sum_{\mathsf{r}}^{\mathsf{R}}\alpha_{\mathsf{r}}\boldsymbol{\sigma}(1-\langle\mathbf{w}^{\mathsf{t}}_{\mathsf{r}},\phi_{\mathbf{x}}\rangle)\bigg)$$

$$\mathfrak{p}\big(\mathsf{y}|\phi_{\mathbf{x}};\mathbf{w}^{\mathsf{t}}_{*}\big) = \mathsf{y}\log\bigg(\sum_{\mathsf{s}}^{\mathsf{R}}\beta_{\mathsf{s}}\boldsymbol{\sigma}(\langle\mathbf{w}^{\mathsf{t}}_{\mathsf{s}},\phi_{\mathbf{x}}\rangle)\bigg) + (1-\mathsf{y})\log\bigg(\sum_{\mathsf{s}}^{\mathsf{R}}\beta_{\mathsf{s}}\boldsymbol{\sigma}(1-\langle\mathbf{w}^{\mathsf{t}}_{\mathsf{s}},\phi_{\mathbf{x}}\rangle)\bigg)$$

$$\text{where } \alpha_{\mathsf{r}} = \frac{\rho^{\mathsf{t}}_{\mathsf{r}}}{\rho^{\mathsf{t}}_{1}+...+\rho^{\mathsf{t}}_{\mathsf{R}}} \quad \beta_{\mathsf{s}} = \frac{e^{\mathsf{logprob}^{\mathsf{t}}_{\mathsf{s}}}}{e^{\mathsf{logprob}^{\mathsf{t}}_{1}}+...+e^{\mathsf{logprob}^{\mathsf{t}}_{\mathsf{R}}}} \quad (\mathsf{r},\mathsf{s}\in 1...\mathsf{R})$$

10: **end if**
11: **return** $\big(\varrho^{\mathsf{t}}, \{\beta_{\mathsf{s}}\}_{\mathsf{s}=1..\mathsf{R}}\big)$

Commentary on $\mathsf{QAudit}_{\varrho}$: it computes the amount of regret value ϱ^{t} that is owed to all relevant Agents at time stage t.

1. In line 1 and 2, $\mathsf{QAudit}_{\varrho}$ computes the data-likelihood of each model w.r.t the test set and validation set. Recall the Auditors have access to each model $\mathbf{w}^{\mathsf{t}}_{\mathsf{k}}$'s individual performance on the validation set at every wagering stage t Moreover, recall by $\mathsf{QAudit}_{\mathbb{D}}$, the test data and training data are drawn from the same distribution, so that the expected ranking is known. In line 7, $\mathsf{QAudit}_{\varrho}$ check if the individual classifier ranking under $\mathbb{D}_{\mathsf{test}}$ is the same as $\mathbb{D}_{\mathsf{val}}$. If they are different then it is due to the natural variation between the test and validation sets, so the Auditors are not penalized.

2. Line 3 computes the best expert ranking, using QAudit_{*} it select the top R models by their individual data-likelihood, and computes the ensemble score. This $\mathsf{logprob}^{\mathsf{t}}_{*}$ is the best the models $\mathbf{W}^{\mathsf{t}}$ could have done at around t, any deviation from this best prediction is *regret*.

3. In line 9, $\mathsf{QAudit}_{\varrho}$ checks that the best expert ensemble in hindsight does do better than the Auditor-selected ensemble, *and* that the Auditors could not have selected this ensemble using the log-likelihood information given.

4. The block of expressions between lines 9-10 computes the regret value denominated in $\mathfrak{miat}$, which is the Bregman divergence between the best ensemble the Auditors have created $\mathbf{w}^{\mathsf{t}}_{\mathsf{Auditor}}$, and the best expert ensemble according to data likelihood ranking $\mathbf{w}^{\mathsf{t}}_{*}$. The intuition is that if the Auditors choose the wrong ensemble, then the protocol

$\mathsf{QAudit}_{\varrho}$ forces the Auditors to play one additional "phantom" stage between t and $\mathsf{t}+1$. In this phantom stage, all the winning Auditors buy shares of the best expert model $\mathbf{w}^{\mathsf{t}}_{*}$, Similar to $\mathsf{QAudit}_{\boldsymbol{\rho}^o}$, this share price is initialized as the Bregman divergence from the Auditor model to the expert model.

Protocol 9 $\mathsf{QAudit}_{\varrho\mathfrak{v}}$ Bregman's regret payment subprotocol

Require: regularization parameter $\mathsf{R} \leq \mathsf{K}/2$, permutation distance function $\mathsf{a} \mid\mid \mathsf{b}$
Require: model parameters $\{\mathbf{w}^{\mathsf{t}}_{\mathsf{k}}\}_{\mathsf{k}=1..\mathsf{K}}$
Require: test set $\mathbb{D}_{\mathsf{test}}$, validation set $\mathbb{D}_{\mathsf{val}}$
Require: price vector for stage t at market close: $\boldsymbol{\rho}^{\mathsf{t}}$, bid history at stage $\mathbf{B}^{\mathsf{t}}$
Require: agent accounts $\{\mathsf{Agent}_{\mathsf{k}}\}_{\mathsf{K}}$, auditor accounts $\{\mathsf{Auditor}_{\mathsf{i}}\}_{\mathsf{N}}$
1: $(\mathsf{logprob}^{\mathsf{t}}_{auditors}, \{\mathbf{w}^{\mathsf{t}}_{\mathsf{r}}\}_{\mathsf{r}=1..\mathsf{R}}) \leftarrow \mathsf{QAudit}_{\mathsf{infer}}(\mathsf{R}, \mathbb{D}_{\mathsf{test}}, \boldsymbol{\rho}^{\mathsf{t}}, \mathbf{W}^{\mathsf{t}})$
2: $(\mathsf{logprob}s, \mathsf{logprob}^{\mathsf{t}}_{*}, \{\mathbf{w}^{\mathsf{t}}_{\mathsf{s}}\}_{\mathsf{s}=1..\mathsf{R}}) \leftarrow \mathsf{QAudit}_{*}(\mathsf{R}, \mathbb{D}_{\mathsf{test}}, \mathbf{W}^{\mathsf{t}})$

3: ▷ *select unpaid best Z agents in hindsight, & Auditors who wagered on winning models*
4: $\{\mathsf{Auditor}_{\mathsf{j}}\}_{\mathsf{j}=1..\mathsf{M}} \leftarrow \mathbf{select}(\{\mathsf{Auditor}_{\mathsf{i}}\}_{\mathsf{i}=1..\mathsf{N}},\ \boldsymbol{\rho}^{\mathsf{t}},\ \mathbf{B}^{\mathsf{t}})$
5: $\{\mathsf{Agent}_{\mathsf{s}}\}_{\mathsf{s}=1..\mathsf{R}} \leftarrow \mathbf{sort}(\{\mathsf{Agent}_{\mathsf{k}}\}_{\mathsf{k}=1..\mathsf{K}},\ \mathsf{logprob}s)[1:\mathsf{R}]$
6: $\{\mathsf{Agent}_{\mathsf{z}}\}_{z=1..Z} \leftarrow \mathbf{subsetDiff}(\{\mathsf{Agent}_{\mathsf{s}}\}_{\mathsf{s}=1..\mathsf{R}}, \boldsymbol{\rho}^{\mathsf{t}})$

7: $\Delta^{\mathsf{t}}_{\mathsf{Agent}} \leftarrow []$ ▷ *regret comp/penalty for Agents*
8: $\Delta^{\mathsf{t}}_{\mathsf{Auditor}} \leftarrow []$ ▷ *regret comp/penalty for Auditors*
9: $(\varrho^{\mathsf{t}}, ...) \leftarrow \mathsf{QAudit}_{\varrho}(\mathsf{a} \mid\mid \mathsf{b}, \mathsf{R}, \mathbf{W}^{\mathsf{t}}, \boldsymbol{\rho}^{\mathsf{t}}, \mathbb{D}_{\mathsf{test}}, \mathbb{D}_{\mathsf{val}})$
10: **if** $\varrho^{\mathsf{t}} > 0$ **then**
11: **for** $\mathsf{Agent}_{\mathsf{z}} \in \{\mathsf{Agent}_{\mathsf{z}}\}_{\mathsf{z}=1..Z}$ **do** ▷ *compensate unpaid agents for regret*
12: let

$$\beta_{\mathsf{z}} = \frac{e^{\mathsf{logprob}^{\mathsf{t}}_{\mathsf{z}}}}{e^{\mathsf{logprob}^{\mathsf{t}}_{1}} + ... + e^{\mathsf{logprob}^{\mathsf{t}}_{\mathsf{z}}}}$$

13: let $\varrho^{\mathsf{t}}_{\mathsf{z}} = \beta_{\mathsf{z}} \cdot \varrho^{\mathsf{t}}$
14: $\mathsf{Agent}_{\mathsf{z}} \leftarrow \mathsf{Agent}_{\mathsf{z}} + \varrho^{\mathsf{t}}_{\mathsf{z}}$
15: $\Delta^{\mathsf{t}}_{\mathsf{Agent}_{\mathsf{z}}} \leftarrow [..., (\mathsf{Agent}_{\mathsf{z}}, \varrho^{\mathsf{t}}_{\mathsf{z}})]$
16: **end for**
17: **for** $\mathsf{Auditor}_{\mathsf{j}} \in \{\mathsf{Auditor}_{\mathsf{j}}\}_{\mathsf{j}=1..\mathsf{M}}$ **do** ▷ *penalize Auditors for regret*
18: $(\mathsf{b}^{\mathsf{t}}_{1,\mathsf{j}}, .., \mathsf{b}^{\mathsf{t}}_{\mathsf{R},\mathsf{j}}) \in \mathbf{B}^{\mathsf{t}}$ ▷ *select auditor j's wager for winning models*
19: $(\mathsf{b}^{\mathsf{t}}_{1,1}, .., \mathsf{b}^{\mathsf{t}}_{\mathsf{R},\mathsf{M}}) \in \mathbf{B}^{\mathsf{t}}$ ▷ *select all winning Auditors' wager for winning models*
20: let $\mathsf{q}^{\mathsf{t}}_{\mathsf{j}} = \frac{\mathsf{b}^{\mathsf{t}}_{1,\mathsf{j}} + ... + \mathsf{b}^{\mathsf{t}}_{\mathsf{R},\mathsf{j}}}{\mathsf{b}^{\mathsf{t}}_{1,1} + ... + \mathsf{b}^{\mathsf{t}}_{\mathsf{R},\mathsf{M}}}$
21: let $\varrho^{\mathsf{t}}_{\mathsf{j}} = \mathsf{q}^{\mathsf{t}}_{\mathsf{j}} \cdot \varrho^{\mathsf{t}}$
22: $\mathsf{Auditor}_{\mathsf{j}} \leftarrow \mathsf{Auditor}_{\mathsf{j}} - \varrho^{\mathsf{t}}_{\mathsf{j}}$
23: $\Delta^{\mathsf{t}}_{\mathsf{Auditor}} \leftarrow [..., (\mathsf{Auditor}_{\mathsf{j}}, \varrho^{\mathsf{t}}_{\mathsf{j}})]$
24: **end for**
25: **end if**
26: **return** $(\varrho^{\mathsf{t}}, \Delta^{\mathsf{t}}_{\mathsf{Agent}_{\mathsf{z}}}, \Delta^{\mathsf{t}}_{\mathsf{Auditor}})$

Commentary on $\mathsf{QAudit}_{\varrho\mathfrak{v}}$:

1. This protocol computes the regret and allocates it to the aggrieved Agents, and penalize the bad Auditors. Line 4-6 pick out the Auditors who selected the winning models, and the Agents who were had "good" models but were not selected by said Auditors. Then in line 12-17, the aggrieved Agents are paid pro-rata to how well their model predicted the likelihood of the test data. Line 19-26 then subtract the pro-rated regret penalty from the Auditors' accounts.

2. Referencing the top level QAudit protocol, $\mathsf{QAudit}_{\varrho\mathfrak{v}}$ is run in two branches of the code, once when the current ensemble improves that of the previous stage, and once if it does not. The notion is that in both cases, there may be aggrieved Agents.

2.4 QAudit Machine Learning

This section introduces the subroutines used to determine the quality of a generative model $\boldsymbol{\theta}_{\mathsf{k}}^{\mathsf{t}}$. This is part of the "tokenization" process whereby a generative model (or any data source) is turned into a machine intelligence nonfungible token ($\mathfrak{mint}$). It proceeds as follows:

1. at every stage t for every model $\boldsymbol{\theta}_1 ... \boldsymbol{\theta}_{\mathsf{K}}$, a corresponding classifier $\mathbf{w}_{\mathsf{k}}^{\mathsf{t}}$ is trained on this model. Each $\mathbf{w}_{\mathsf{k}}^{\mathsf{t}}$ measures the quality of the current generative model snapshot w.r.t some PA-defined ground truth. This training is done via *online logistic regression* [ZJ+12], which produces classifier versions: $\left\{\mathbf{w}_{\mathsf{k}}^{1}, \ldots, \mathbf{w}_{\mathsf{k}}^{\mathsf{T}+1}\right\}$. In online regression, the model is not trained to completion at each stage. Instead the learner samples a few examples from the corresponding $\boldsymbol{\theta}_{\mathsf{k}}$, and updates the regression model.[8]

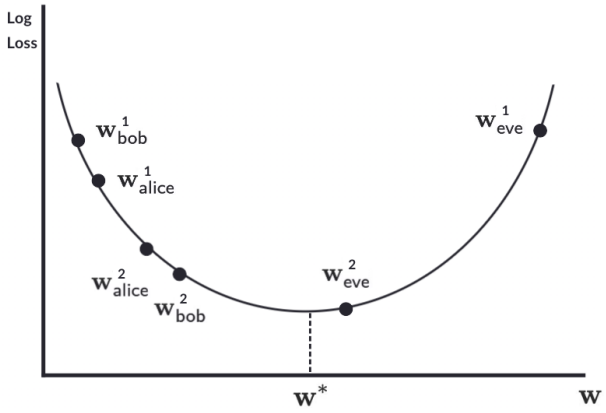


Figure 2.10: QAudit trains one shared instance of logistic regression model from the underlying generative models $\boldsymbol{\theta}_{\mathsf{k}}^{1} ... \boldsymbol{\theta}_{\mathsf{k}}^{\mathsf{T}}$, and updates the model in an online manner over all Agents $\mathsf{k} = 1..\mathsf{K}$. At stage $\mathsf{t} = 1$ in the diagram above, Agents alice, bob and eve all submit generative models $\left\{\boldsymbol{\theta}_{\mathsf{alice}}^{1}, \boldsymbol{\theta}_{\mathsf{bob}}^{1}, \boldsymbol{\theta}_{\mathsf{eve}}^{1}\right\}$. Data is sampled from these models and used to train classifiers $\left\{\mathbf{w}_{\mathsf{alice}}^{1}, \mathbf{w}_{\mathsf{bob}}^{1}, \mathbf{w}_{\mathsf{eve}}^{1}\right\}$. At stage $\mathsf{t} = 2$, the classifiers sample from the generative models again and updated to create versions $\left\{\mathbf{w}_{\mathsf{alice}}^{2}, \mathbf{w}_{\mathsf{bob}}^{2}, \mathbf{w}_{\mathsf{eve}}^{2}\right\}$. This process continues for each stage, so that the Agents are essentially racing down the same log-loss hill to see who can reach the bottom point $\mathbf{w}^*$ the fastest.

2. Model share $\mathfrak{mint}$s are issued from every $\mathbf{w}_{\mathsf{k}}^{\mathsf{t}}$. Each $\mathfrak{mint}_{\mathsf{k}}^{\mathsf{t}}$ is a statement that the underlying generative model $\boldsymbol{\theta}_{\mathsf{k}}$ is better than other generative models in the same stage t, *and* better than generative models in stage $\mathsf{t}-1$. Over the course of one QAudit instance, at least $(\mathsf{T}+1) \times \mathsf{K}$ $\mathfrak{mint}$s will be issued.

3. In order to prevent the classifier model from saturating, QAudit periodically re-initializes all the regression models. This is known as *regime changes*, and is defined in 2.6.8.

The advantages of scoring a regression surrogate as part of the tokenization process instead of the underlying generative model are:

[8]The introduction stated that the overall setup of $\mathsf{QAudit}_{\mathsf{train}}$ and $\mathsf{QAudit}_{\mathsf{infer}}$ may superficially resemble generative adversarial networks (GANs) of [GP+14], however the rewards of the generator $\mathfrak{p}(\phi_{\mathbf{x}_\tau}|\phi_{\mathbf{x}_\tau}; \boldsymbol{\theta})$ is not tied to that of the discriminator $\mathfrak{p}(\mathsf{y}|\phi_{\mathbf{x}}; \mathbf{w})$. So online regression is the right comparison here.

1. (*incentive compatibility*): logistic regression have convex log-loss functions as seen in E.3.3. A convex loss-function allows QAudit to place all $(\mathsf{T}+1)\times\mathsf{K}$ models onto the same cost curve for comparison. It also ensures the scoring-rule we use to compute the quality of models across versions is incentive compatible, as expressed in D.2. See 2.7 and 2.10 for visual demonstration.

2. (*easier go-to-market*): scoring surrogates allow QAudit markets to operate even if the underlying model is not actively updated. This is an important point, model-trainers may not push checkpoints within every stage t. Moreover, there are many open-source foundational models whose parameters are frozen. Metheus can bootstrap from this reservoir of free models by tokenizing them. This negates the need to launch Metheus with a set of human competitors updating live models, which may be prohibitively expensive to collate.

3. (*broader use case*): the underlying generative model may not be a model at all, but instead a database of user-generated data. This allows QAudit to run a data market that judges the quality of data as they are streamed into the Metheus data-lake.

Protocol 10 $\mathsf{QAudit}_{\mathsf{train}}$ training discriminative surrogate subroutine

Require: model parameter $\boldsymbol{\theta}$, noise generator $\boldsymbol{\eta}$,
Require: discriminative model parameter $\mathbf{w}_{\mathsf{k}}^{\mathsf{t}-1}$ from last stage
Require: data set $\mathbb{D}_{\mathsf{val}}$, public query set $\mathbb{Q}$, embedding ϕ, number of samples m

$\mathbb{D}_{\mathsf{train}} \leftarrow \{\}$
for $\mathsf{i} \in 1...\mathsf{m}$ **do** ▷ *build training set*
  $(\mathbf{z}, \mathbf{x}) \leftarrow (\boldsymbol{\eta}(), \mathfrak{p}(\mathbf{x}|\mathbf{x}';\boldsymbol{\theta}))$ where $\mathbf{x}' \in \mathbb{Q}$
  $\mathbb{D}_{\mathsf{train}} \leftarrow [..., (\phi_{\mathbf{x}}, 1), (\phi_{\mathbf{z}}, 0)]$
end for
scramble$(\mathbb{D}_{\mathsf{train}})$
$\mathbf{w}_{\mathsf{k}}^{\mathsf{t}} \leftarrow \mathsf{trainLogisticRegression}(\mathbb{D}_{\mathsf{train}}, \mathbf{w}_{\mathsf{k}}^{\mathsf{t}-1})$ ▷ *update model on* $\mathbb{D}_{\mathsf{train}}$
let $\mathsf{logprob}_{\mathsf{k}}^{\mathsf{t}} =$

$$\sum_{(\phi_{\mathbf{x}},y)\in\mathbb{D}} \log \mathfrak{p}(\mathsf{y}|\phi_{\mathbf{x}};\mathbf{w}_{\mathsf{k}}^{\mathsf{t}}) \text{ where}$$

$$\log \mathfrak{p}(\mathsf{y}|\phi_{\mathbf{x}};\mathbf{w}_{\mathsf{k}}^{\mathsf{t}}) = \mathsf{y}\cdot\log\boldsymbol{\sigma}(\langle\mathbf{w}_{\mathsf{k}}^{\mathsf{t}},\phi_{\mathbf{x}}\rangle) + (1-\mathsf{y})\cdot\log\boldsymbol{\sigma}(1-\langle\mathbf{w}_{\mathsf{k}}^{\mathsf{t}},\phi_{\mathbf{x}}\rangle)$$

return $(\mathbf{w}^{\mathsf{t}}, \mathsf{logprob}_{\mathsf{k}}^{\mathsf{t}})$

Commentary on $\mathsf{QAudit}_{\mathsf{train}}$:

1. it trains a kernel logistic regression classifier $\mathbf{w}_{\mathsf{k}}^{\mathsf{t}}$ (see E.3.3) by sampling from generative model $\boldsymbol{\theta}_{\mathsf{k}}^{\mathsf{t}}$, or data source $\mathbb{D}_{\mathsf{k}}^{\mathsf{t}}$. This model $\mathbf{w}_{\mathsf{k}}^{\mathsf{t}}$ is updated on every stage t, so that over the T stages $\mathsf{QAudit}_{\mathsf{train}}$ is doing *online* kernel logistic regression [ZJ+12]. The parameters that control the confidence of $\mathsf{QAudit}_{\mathsf{train}}$ are sample size m, embedding ϕ, and negative example or noise generator $\boldsymbol{\eta}$.

2. (*negative example generator* $\boldsymbol{\eta}$): this generator may emit noise $\mathbf{z}$ that is an appropriate control data against the model $\boldsymbol{\eta}$. This negative example is labeled with $\mathsf{y}=0$ to form $(\mathbf{z}, 0)$, it is fed to $\mathbf{w}_{\mathsf{k}}^{\mathsf{t}}$ as an example of "bad data." Note the noise generator could in fact be an earlier example of a generative model. Or if the data source $\mathbb{D}_{\mathsf{k}}^{\mathsf{t}}$ is a pool of naturally occurring data within a class, then $\boldsymbol{\eta}$ could generate images outside of this class. For example if $\mathbb{D}_{\mathsf{k}}^{\mathsf{t}}$ emit images of apples, then $\boldsymbol{\eta}$ should emit images of oranges.

3. (*sample size* m): controls how many examples $\mathsf{QAudit}_{\mathsf{train}}$ should sample from $\boldsymbol{\theta}$. The greater than m, the more confident that the classifier is accurately determining how

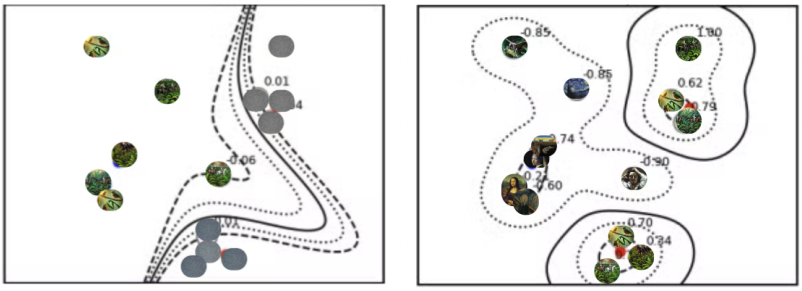


Figure 2.11: Two variations of $\mathsf{QAudit}_{\mathsf{train}}$ under proposal. In the left variation, Agent generative image models $\boldsymbol{\theta}$'s. Images sampled from said models form a training set against a control noise generator $\boldsymbol{\eta}$. This dataset is then used to train a discriminative model $\mathfrak{p}(\mathsf{y}|\mathbf{x};\boldsymbol{\theta})$ that identify noise from image. On the right, images sampled from generative models is trained against true images drawn by real painters. This corresponds to a classifier that can discriminate machine generated images from real ones. Observe that depending on which variation of the tokenization process $\mathsf{QAudit}_{\mathsf{train}}$ uses, the ranking over models differ.

well $\boldsymbol{\theta}$ has improved $\mathbf{w}$. However, this also may lead to faster convergence for the classifier, thus leading to lesser improvements during the later times. If the data source is just one datum, as may be the case if the Agent is simply selling a picture or a song, then the noise generator $\boldsymbol{\eta}$ must only emit one out of class negative example as well.

4. (*embedding* ϕ): from a highly reductionist perspective, machine learning is fundamentally about function search inside a space defined entirely by the data. Thus the manner with which this data is represented determines how the function is constructed, see E.3. The embedding function ϕ ingests raw data and outputs a (potentially high dimensional) representation of this data. They may be drawn from $\mathbb{D}_{\mathsf{val}}$, $\mathbb{D}_{\mathsf{test}}$, $\boldsymbol{\theta}$, or $\mathbb{D}$. The choice of ϕ could significantly affect the outcome of which model is "better."

Protocol 11 $\mathsf{QAudit}_{\mathsf{infer}}$ ensemble inference subroutine

Require: regularization parameter $\mathsf{R} \leq \mathsf{K}/2$, data set $\mathbb{D}$
Require: price vector for stage t at market close: $\boldsymbol{\rho}^{\mathsf{t}}$
Require: model parameters $\mathbf{W}^{\mathsf{t}}$
1: $\{\mathbf{w}_{\mathsf{r}}^{\mathsf{t}}\}_{\mathsf{r}=1..\mathsf{R}} \leftarrow \mathbf{sort}\big(\mathbf{W}^{\mathsf{t}}, \{\rho_{\mathsf{k}}^{\mathsf{t}}\}_{\mathsf{k}=1..\mathsf{K}}\big)[1:\mathsf{R}]$ ▷ *select top* R *models*
2: let $\mathsf{logprob}^{\mathsf{t}} = \sum_{(\phi_{\mathbf{x}},y)\in\mathbb{D}_{\mathsf{test}}} \log \mathfrak{p}(\mathsf{y}|\phi_{\mathbf{x}};\mathbf{w}^{\mathsf{t}})$ where

$$\begin{aligned}\log\mathfrak{p}(\mathsf{y}|\phi_{\mathbf{x}};\mathbf{w}^{\mathsf{t}}) &= \log\mathfrak{p}(\mathsf{y}|\phi_{\mathbf{x}};\{\mathbf{w}_{\mathsf{r}}^{\mathsf{t}}\}_{\mathsf{r}=1..\mathsf{R}}) \\ &= \mathsf{y}\log\mathfrak{p}(\mathsf{y}=1|\phi_{\mathbf{x}};\mathbf{w}^{\mathsf{t}}) + (1-\mathsf{y})\log\mathfrak{p}(\mathsf{y}=0|\phi_{\mathbf{x}};\mathbf{w}^{\mathsf{t}}) \\ &= \mathsf{y}\log\left(\sum_{\mathsf{r}}^{\mathsf{R}}\alpha_{\mathsf{r}}\boldsymbol{\sigma}(\langle\mathbf{w}_{\mathsf{r}}^{\mathsf{t}},\phi_{\mathbf{x}}\rangle)\right) + (1-\mathsf{y})\log\left(\sum_{\mathsf{r}}^{\mathsf{R}}\alpha_{\mathsf{r}}\boldsymbol{\sigma}(1-\langle\mathbf{w}_{\mathsf{r}}^{\mathsf{t}},\phi_{\mathbf{x}}\rangle)\right)\end{aligned}$$

$$\text{and } \alpha_{\mathsf{r}} = \frac{\rho_{\mathsf{r}}^{\mathsf{t}}}{\rho_1^{\mathsf{t}} + ... + \rho_{\mathsf{R}}^{\mathsf{t}}}$$

3: **return** $\big(\mathsf{logprob}^{\mathsf{t}}, \{\mathbf{w}_{\mathsf{r}}^{\mathsf{t}}\}_{\mathsf{r}=1..\mathsf{R}}\big)$

$\mathsf{QAudit}_{\mathsf{infer}}$ selects the winning R classifier models, and use the weighted ensemble of $\mathbf{w}_{\mathsf{r}}^{\mathsf{t}}$'s to predict the label y given some input $\phi_{\mathbf{x}}$ drawn from the test set $\mathbb{D}_{\mathsf{test}}$. The weight is determined by the final price vector $\boldsymbol{\rho}^{\mathsf{t}}$ at market close on stage t. Thus the Auditors control how each model $\mathbf{w}_{\mathsf{r}}^{\mathsf{t}}$ is weighed.

Remark 2.4.1. (*Protocol Composability*) QAudit can compose with other machine learning protocols for bespoke tokenization process.

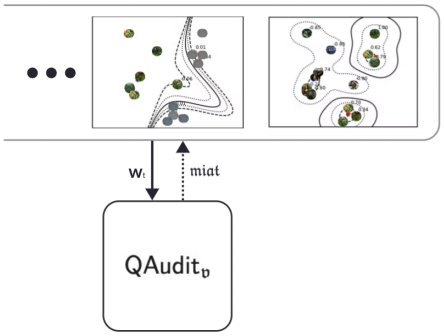


Figure 2.12: Given a fixed $\mathsf{QAudit}_{\mathfrak{v}}$ wager-elicitation module, we can pair it with any number of $\mathsf{QAudit}_{\mathsf{train}}$ machine learning modules. The different modules should be developed by the open source community, who will naturally have different metrics as to what a "good" model is. Critically, as different QAudit market instances use the metric modules, the development team behind them are compensated by miat.

This arrangement is ideal as there is no one way to judge models, so a "market place of metrics" is the most value-neutral resolution. Additionally, multiple tokenization modules also enable Metheus to launch multiple QAudit markets *per* set of generative models.

2.5 QAudit Elementary Analysis

QAudit is a multi-faceted market whereby Agents are bartering for miat with models, while the Auditors are bidding for miat with miat. In every pair of stages $(\mathsf{t}, \mathsf{t}+1)$ where t is odd and $\mathsf{t}+1$ is even, the Agent move first by reporting their model, and then the Auditors move next to wager on the models. The competitive nature of QAudit may be summarized as a game amongst four sets of adversaries:

1. at every stage t, Agents are competing against all other Agents for the top R positions.
2. At every stage t, each Agent is competing against all other $\mathsf{R}-1$ Agents for a share of the bottom $\mathsf{K}-\mathsf{R}$ Agents' associated wagers.
3. At each stage t, Auditors are competing against other Auditors for 1) the top R slots with the highest wager amount, and 2) a percentage of the wagering amount within each of the R slots from $\mathsf{t}+1$.
4. Over the course of all t stages, every generative model $\boldsymbol{\theta}_{\mathsf{k}}^{\mathsf{t}}$ or data source $\mathbb{D}_{\mathsf{k}}^{\mathsf{t}}$ is in a min-max game with a corresponding discriminative classifier $\mathbf{w}_{\mathsf{k}}^{\mathsf{t}}$. In the first case, the generative model seek to spoof the classifier that it is emitting data "sampled from nature," while the classifier is attempting to detect fraudulent model generated data. In the second case, the Agent-provided data base $\mathbb{D}_{\mathsf{k}}^{\mathsf{t}}$ is attempting to confuse the classifier that it is emitting data from the correct class, while the classifier is attempting to detect out of class data.

The QAudit protocol features a symphony of incentives, with many games interleaved concurrently:

1. The sub-protocols QAudit_{Θ}, $\mathsf{QAudit}_{\mathfrak{v}}$, and $\mathsf{QAudit}_{\mathsf{infer}}$ are running online learning with mixture of experts using "follow the regularized leader" (FTRL) algorithm.
2. $\mathsf{QAudit}_{\mathsf{train}}$ is running online kernel logistic regression by sampling from models $\{\boldsymbol{\theta}_{\mathsf{k}}^{\mathsf{t}}\}_{\mathsf{k}=1..\mathsf{K}}$ or data sources $\{\mathbb{D}_{\mathsf{k}}^{\mathsf{t}}\}_{\mathsf{k}=1..\mathsf{K}}$ over time stages $\mathsf{t}=1...\mathsf{T}+1$.
3. The Dutch auction function $\mathsf{v}_{\rho}^{\mathsf{t}}$ is running ascending price auction for model shares $\mathsf{k}=1...\mathsf{K}$ over all stages $\mathsf{t}=1...\mathsf{T}+1$.

In lieu of these forces, we cue the arguments of [CV10] and [ACV12], and analyze QAudit as 1) a prediction market, 2) a game, and 3) a randomized algorithm.

1. In the first view, QAudit has market participants who report information, which must be collated to form a "correct" view of the world.
2. In the second view, QAudit has Auditors that must be incentivized to settle on equilibrium points. These points must be of "positive welfare" from the perspective of Agents.
3. In the third view, QAudit has access to randomness due to unobserved player motivations, and must find the correct solution in "reasonable amount of time."

Now given this perspective, this section characterizes QAudit with these properties:

1. lemma 2.5.1: QAudit reward structure is *incentive compatible* from the perspective of the Agents. Thus reporting the truth is a *pure strategy equilibrium* (lemma 2.5.2).
2. lemma 2.5.3: collusion, defined as Agents who report the best model, and Auditors who wager on the best model, is a *mixed-strategy equilibrium*.
3. lemma 2.5.8: in the event where every $\mathsf{Auditor}_\mathsf{i}$ accepts the prompted rankings $\mathfrak{s}_\mathsf{r}^*$ provided by QAudit instance, then the Auditors will collectively find the correct permutation in $\mathsf{poly}(\mathsf{K!})$ number of wagers.
4. lemma 2.5.10: in the event where some of the Auditors hold private belief differing from the market maker's suggestion, then the idealized distribution $\tilde{\boldsymbol{\delta}}_\mathsf{r}$ over all K! rankings degrades by:

$$\left\|\mathbf{p} - \tilde{\boldsymbol{\delta}}_\mathsf{r}\right\|_1 \leq \frac{2}{1-\alpha_\mathsf{r}}|\Delta\alpha_\mathsf{r}| \quad \|\tilde{\boldsymbol{\delta}}_\mathsf{r} - \mathbf{p}\|_1 \leq (1-\alpha_\mathsf{r})\|\Delta\mathbf{e}\|_1 \quad \|\tilde{\boldsymbol{\delta}}_\mathsf{r} - \mathbf{p}\|_1 \leq \frac{\alpha_\mathsf{r}}{1-\alpha_\mathsf{r}}\|\Delta\mathbf{M}_\mathsf{r}\|_\infty.$$

   Stating the degradation in the QAudit solution as a function of:

   a) proportion $1-\alpha_\mathsf{r}$ of Auditors who hold differing opinion.
   b) The Auditors' private preference over rankings denoted $\mathbf{e} + \Delta\mathbf{e}$.
   c) How aggressively the Auditors wager, as expressed in the PSA walk matrix $\mathbf{M}$.

5. lemma 2.5.11: if Auditors are partitioned into two blocks, so that some of the Auditors accept the prompted permutation, while others hold a different opinion. Then if the proportion of Auditors who hold private belief $\mathfrak{s}_\mathsf{r}^*$ is greater than 50%, the likelihood of finding one of the two accepted solution states is greater than $\frac{1}{2}$.

Ultimately, the soundness of QAudit is built upon these fundamental theorems:

1. The QAudit *Consensus Theorem* (2.6.4) states that for every stage t, there exists a *Correlated Equilibrium* (see definition B.2.3) s.t. the market consensus at market close reflects the true permutation $\mathfrak{s}_\mathsf{r}^{*\mathsf{t}}$ in expectation, provided the permutation is prompted by $\mathsf{QAudit}_\mathfrak{o}$ at market open in every t.
2. The QAudit *Social Consensus Theorem* (2.6.6) states that in the event where the protocol suggests many possible rankings over models, denoted $\mathfrak{s}_1^*, \ldots, \mathfrak{s}_m^*$. Then in every protocol stage t, there exists a point in time τ^*:

$$\tau_\mathsf{f} - \mathsf{N} - \mathsf{K}\log(\mathsf{K}) \leq \tau^* < \tau_\mathsf{f} - 1 - \mathsf{K}\log(\mathsf{K}),$$

   s.t. it is rational for every Auditor to switch to wagering on a single ranking $\mathfrak{s}_\mathsf{r}^*$, favoring this ranking over all others.
3. QAudit *is not a Loser's Game* (2.6.10): states that the QAudit sequential parimutuel reward economic structure has positive expected value for some Auditor - if he plays according to rankings $\mathfrak{s}_\mathsf{r}^*$ prompted by the protocol.

The rest of this chapter introduces the necessary computational models, definitions, and concepts. It then walks through the lemmas at a high level, all the proofs are found in Appendix C.

Agent-Agent Sub-Game

We can make some assumptions on the behavior of players:

1. Agent*s* maximize their earnings on every stage t against other Agents, independent of what occurred in the previous or next stages, conditioned on the previous stage $\mathsf{t}-1$ (Markov Independence).

2. Auditor*s* are myopic with 3-stage time horizon. They only consider the stages $(\mathsf{t}-1, \mathsf{t}, \mathsf{t}+1)$, and seek to maximize profit subject to the cost of acquiring winnings from the next stage $\mathsf{t}+1$, or the mining reward at $\mathsf{T}+1$. Moreover:

   a) since the Auditor cannot determine the amount wagered over the top R models in stage $\mathsf{t}+1$, he can only assume it is some constant.

   b) Markov independence: the Auditor can inspect the model in stage $\mathsf{t}-1$. But Auditors' wagers are independent of model performance in stages $\mathsf{t}=1...\mathsf{T}-2$, conditioned on the model performance in the previous stage $\mathsf{t}-1$.

The next two lemmas assume Agents submit the discriminative classifier weights $\mathbf{w}$, instead of generative model $\boldsymbol{\theta}$ or data set $\mathbb{D}$.

Lemma 2.5.1. Agent *reward* $\mathfrak{v}_r^{\mathsf{t}}$ *is incentive compatible if* Agent*s are myopic and Markov. So that if* Agent_r *has two private models* $\mathbf{w}_r^{\mathsf{t}}$ *and* $\mathbf{v}_r^{\mathsf{t}}$ *with:*

$$\prod_{(\phi_{\mathbf{x}},\mathsf{y})\in\mathbb{D}} \mathfrak{p}(\mathsf{y}|\phi_{\mathbf{x}};\mathbf{w}_r^{\mathsf{t}}) > \prod_{(\phi_{\mathbf{x}},\mathsf{y})\in\mathbb{D}} \mathfrak{p}(\mathsf{y}|\phi_{\mathbf{x}};\mathbf{v}_r^{\mathsf{t}}), \tag{2.1}$$

then the rational Agent_r *will reveal the better model* $\mathbf{w}_r^{\mathsf{t}}$ *under the payment defined in* $\mathsf{QAudit}_{\mathfrak{v}}$.

Proof. See proof in Appendix C.1. ■

Lemma 2.5.2. *Reporting the truth is a pure strategy equilibrium for every* Agent. *That is to say given two* Agent_r *and* Agent_s*, whereby if each* Agent *can report the truth* $\mathbf{w}^{\mathsf{t}}$*, or a lesser model* $\mathbf{v}^{\mathsf{t}}$*, then every* Agent *will report* $\mathbf{w}^{\mathsf{t}}$ *regardless of how the other* Agent *plays.*

Proof. See Appendix C.1. ■

Informally, the reason the Agent is incentivized to report his best model is because he is sharing the wagering pool with other winning Agents. And his reward is weighed not by the price $\rho_{\mathsf{r}}^{\mathsf{t}}$ of his model as determined by the Auditors, but rather by the model's data likelihood as controlled by himself. So the Agent may rely on the Auditors to make the top R slots, but once he has cleared the hurdle he controls his own rewards. This is the principle of "**eat what you bartered with.**"

Agent-Auditor Sub-Game

This section clarifies the intuition on why Agents and Auditors should collude to submit and then wager on the best model.

Lemma 2.5.3. *Collusion is a mixed-strategy equilibrium (see definition B.2.2) between* Agent *and* Auditor. *In this case "collusion" informally refers to the phenomenon whereby the* Agent*s report their best model in some stage* t*, and the* Auditor*s wager on the best reported model in this stage. Moreover if two models* $\mathbf{w}_r$ *and* $\mathbf{w}_s$ *are the best two models, then if expected data likelihood under* $\mathbf{w}_r$ *is greater than* $\mathbf{w}_s$*, then* Auditor*s should wager more on* $\mathbf{w}_r$ *than* $\mathbf{w}_s$.

Proof. See Appendix C.2. ■

Informally, collusion occurs because Agents and Auditors are profiting from different wagering pools, as long as the Auditor does not defect. So Agents fish from the tth stage pool, while Auditors hunt the pool from the $(t+1)$th stage. This is critical, Agents and Auditors are not competing but colluding: their incentives are aligned and their rewards are orthogonal. This is the principle of "**hunt together but eat separately.**"

In reality, QAudit is *not* presented as a competition to the Agents. Instead, it is a subsidized training environment, whereby in exchange for lower training/server costs, the Agents allow their checkpoints to be tokenized and traded. The Agents are tangentially aware of this, but not necessarily informed by this competitive dynamic every time they submit a checkpoint. An analogy can be made with FANNG employees, whose stock-based compensation make up a significant portion of their overall package. Technically the FANNG employee is negotiating their bonuses with traders every day on the job. But this underlying reality does not inform their daily behavior. Similarly, the Agent should see QAudit as a "far away market." If they do plan against it, ideally they do not think deeper than what is written here.

In light of this knowledge, the Auditors should by virtue of the larger Metheus software offering, trust that the Agents are not gaming the system, but rather focused on doing good work. This is why there is no Auditor-Agent sub-game. That is to say if the Agents observe the Auditors wager on a certain kind of models in stages $[t-2, t-1, t]$, should the Agents submit a different model on stage $t+1$. If the Agents allow Auditors' behavior to bias their work, then the solution is not to change the incentives of QAudit, but rather to design a better user interface that further abstracts the market away from them.

Auditor-Auditor Sub-Game within every Stage

While the Agents are bartering for miat with their models, the Auditors are buying miat (read: money) directly with money. This arrangement may appear simple, however Auditor-Auditor dynamic is a complex one: featuring a phase change from cooperative to competitive paradigm depending on market state. So it is not necessarily a given that the Auditors will settle on the correct ranking $\mathfrak{s}^*$ over the Agents' models. For example, although proof 2.5.3 shows that Auditors should prefer the better model reported across Agents, should they decide to "play conservatively" by waiting for other Auditors to move first, then the market may close before the Auditors bid up the best models. We will show that identifying the best models in a timely manner is possible. The rest of this section develops the basic tools and prove some elementary results, and builds towards the fundamental theorem of QAudit stated in the sequel.

First, it is helpful to review how an Auditor would make his buy decision if QAudit is a vanilla sequential parimutuel prediction market of Appendix D.2. Similar to QAudit, the market keeps a persistent inventory state $\mathbf{s}^t = (s_1^t, \ldots, s_K^t)$ over K Agents. Then logarithmic market scoring rule (LMSR) computes a price vector $\boldsymbol{\rho}^t = (\rho_1^t, ..., \rho_K^t)$ from the shares, where each ρ_k^t is computed with:

$$\rho_k^t = \frac{\exp(s_k^t)}{\exp(s_1^t) + \ldots + \exp(s_K^t)} \quad s_k^t \geq 0, \quad 0 \leq \rho_k^t \leq 1.$$

In the vanilla market, the Auditors approach the market sequentially and individually, and he is responsible for his own loss and rake the entire winnings if his buy signal improves the market estimate of the probability or price vector $\boldsymbol{\rho}^t$. Thus if the rational Auditor_i with private estimate $\mathfrak{q}_i = (q_{i,1}, ..., q_{i,K})$ observes some item r is under-priced, then he should report a buy vector $\mathbf{b}_i^t = (0.., b_{i,r}^t, ..0)$ so that the market price ρ_r^t matches his private estimate $q_{i,r}$. That is he reports buy-signal for some share $b_{i,r}^t$ by solving:

$$\rho_r^t = q_{i,r}^t = \frac{\exp(s_r^t)\exp(b_{i,r}^t)}{\exp(s_r^t)\exp(b_{i,r}^t) + \sum_{k\in\{1..K\}/r}\exp(s_k^t)} \implies b_{i,r}^t = \log\left(\frac{q_{i,r}^t \cdot \sum_{k\in\{1..K\}/r}\exp(s_k^t)}{\exp(s_r^t)(1-q_{i,r}^t)}\right). \tag{2.2}$$

Once he buys $\mathsf{b}^{\mathsf{t}}_{\mathsf{i},\mathsf{r}}$ shares, the market price vector is updated to $\boldsymbol{\rho}^{\mathsf{t}+1}$. Then the $\mathsf{Auditor}_{\mathsf{i}}$'s compensation would be how well his report improved the data likelihood under some data set $\mathbb{D}_{\mathsf{test}}$. So pick some $(\mathbf{x}, \mathsf{y}) \in \mathbb{D}_{\mathsf{test}}$, $\mathsf{Auditor}_{\mathsf{i}}$ is rewarded with:

$$\log \mathfrak{p}(\mathsf{y}|\mathbf{x}; \boldsymbol{\rho}^{\mathsf{t}+1}) - \log \mathfrak{p}(\mathsf{y}|\mathbf{x}; \boldsymbol{\rho}^{\mathsf{t}}),$$

at reward time as eqn D.12 states. Recall section 2.2 lists the shortcomings of this simplistic setup. The QAudit protocol fixes these shortcomings, but introduces additional competitive dynamics. Since the Auditors are competing with other Auditors for winnings, he must modulate the degree with which he bids the entire truth across time increments τ in each stage t. And weighs his private truth against:

1. how likely will his report tip the balance and place the Agent in the top R slots.
2. How likely will this Agent *stay* in the winning slots once there. That is the price of this action relative to risk of buying $\mathsf{Agent}_{\mathsf{r}}$, only to be swamped by other Auditors who buy some $\mathsf{Agent}_{\mathsf{s}}$ who subsequently crowd out $\mathsf{Agent}_{\mathsf{r}}$.
3. The size of the wager on stage $\mathsf{t}+1$ that would reward his desired slot at stage t.

Despite this complex dynamic, we show that under certain conditions, reporting the truth is a *Correlated Equilibrium* (see definition B.2.3). However, the price signal $\boldsymbol{\rho}^{\mathsf{t}}$ alone is no longer a fair reflection of the Agents' true beliefs, as they may bid up the price as the market closes. So instead of analyzing the market price vector, we will examine the underlying value of interest: the permutation over K Agents, denoted with $\mathfrak{s}$. The next few definitions will introduce the computational concepts to codify the setting, namely:

1. a rational *audit strategy* corresponding to how the Auditors wager on the Agents.
2. A *permutation state machine* that tracks how the permutation over K Agents change as a response to this strategy while the market is open.

The purpose of this analysis is to state that if every players play the rational audit strategy using a permutation $\mathfrak{s}^*_{\mathsf{r}}$ prompted by QAudit, then $\mathsf{QAudit}_{\mathfrak{v}}$ will converge onto this best permutation in expectation in every stage t.

Remark 2.5.4. (*The Card Shuffle Problem*) The reduced Auditor-Auditor game is analogous to the card shuffling problem: given one deck of cards whereby each card correspond to one Agent, and a table of player so that each player is one Auditor. Let each Auditor have some private belief stating that there is some optimal ordering of the cards, corresponding to the best ordering of Agents' models on stage t. And let this private belief be influenced by some ideal permutation provided by the dealer. Now the game commences, and the deck is passed around the table of Auditors, and each Auditor may switch the position of some cards, rearrange the entire deck, or do nothing. Two questions are asked at the end of the game: 1) what is the distribution over all 52! possible arrangements of cards, and 2) if some or all of the auditors believe in the optimal prompted arrangement, how many times does the deck have to be passed around to reach this best arrangement. See [LPW17, p. 100] for a variation of this problem.

Definition 2.5.5. (*Audit Strategy*) for some $\mathsf{Auditor}_{\mathsf{i}}$ is a sequence of buy vectors:

$$\begin{aligned} &\mathsf{audit\ strategy} = \left[\mathbf{b}^{\mathsf{t}}_{\mathsf{i},\tau_{\mathsf{o}}}, \dots \mathbf{b}^{\mathsf{t}}_{\mathsf{i},\tau_{\mathsf{f}}}\right] \\ &\quad \text{where } \mathbf{b}^{\mathsf{t}}_{\mathsf{i},\tau} = \left(\mathsf{b}^{\mathsf{t}}_{\mathsf{i},1,\tau}, \dots, \mathsf{b}^{\mathsf{t}}_{\mathsf{i},\mathsf{K},\tau}\right) \quad \mathsf{s.t.}\ \mathsf{b}^{\mathsf{t}}_{\mathsf{i},\mathsf{k},\tau} \geq 0, \end{aligned} \tag{2.3}$$

whereby the value of every buy vector $\mathbf{b}^{\mathsf{t}}_{\mathsf{i},\tau}$ is conditioned on the Auditor's private belief, and the current market state. Note each buy vector $\mathbf{b}^{\mathsf{t}}_{\mathsf{i},\tau}$ is indexed by time increments $\tau_{\mathsf{o}} \leq \tau \leq \tau_{\mathsf{f}}$ within stage t. Denoting that at time τ within wagering stage t as defined by $\mathsf{QAudit}_{\mathfrak{v}}$, $\mathsf{Auditor}_{\mathsf{i}}$ buys $\mathsf{b}^{\mathsf{t}}_{\mathsf{i},\mathsf{r},\tau}$ shares of $\mathsf{Agent}_{\mathsf{r}}$'s model. [9]

[9] Recall a model share is some nonfungible machine intellgience token $\mathfrak{mint}^{\mathsf{t}}_{\mathsf{r}}$.

Normally we do not write the audit strategy in sequence form, but rather in "algorithm" form. So it is akin to an automated trading bot. In particular, AuditStrat is the response of interest for the remaining section, we assume it is used by Auditors while trading against the market maker $\mathsf{QAudit}_{\mathfrak{v}}$.

Algorithm 12 $\mathsf{AuditStrat}(\mathfrak{s}_{\mathsf{r}}^{*}, C, \mathsf{d})$: Audit Strategy for every $\mathsf{Auditor}_{\mathsf{i}}$

Require: QAudit's suggested permutation $\mathfrak{s}_{\mathsf{r}}^{*}$
Require: $\mathsf{Auditor}_{\mathsf{i}}$'s fixed private belief $\mathfrak{q}_{\mathsf{i}}^{\mathsf{t}} = \left(\mathsf{q}_{\mathsf{i},1}^{\mathsf{t}}, \ldots, \mathsf{q}_{\mathsf{i},\mathsf{K}}^{\mathsf{t}}\right)$.
Require: $\mathsf{Auditor}_{\mathsf{i}}$'s fixed competitive auction parameters $C \geq 0$, $\mathsf{d} \geq 0$.

$\mathfrak{s}_{\mathsf{i}}^{\mathsf{t}} \leftarrow \mathsf{induceOrdering}\left(\mathfrak{q}_{\mathsf{i}}^{\mathsf{t}}\right)$ OR $\mathfrak{s}_{\mathsf{r}}^{*}$ ▷ Auditor *can accept or reject protocol suggestion*
while market open **do**
  $\boldsymbol{\rho}_{\tau}^{\mathsf{t}} \leftarrow \mathsf{QAudit}_{\mathfrak{v}}$
  $\mathfrak{s}_{\tau}^{\mathsf{t}} \leftarrow \mathsf{induceOrdering}\left(\boldsymbol{\rho}_{\tau}^{\mathsf{t}}\right)$
  for $\mathsf{Agent}_{\mathsf{r}} \in \mathfrak{s}_{\tau}^{\mathsf{t}}$ **do**
    if $\mathsf{r} \in \mathfrak{s}_{\mathsf{i}}^{\mathsf{t}}[1..\mathsf{R}]$ & $\mathsf{r} \leq \mathsf{R}$ **then** ▷ Agent_{r} *is already ranked in top* R
      Flip coin:
      if Head **then**
        buy $C \cdot \mathsf{b}_{\mathsf{r},\mathsf{i}}^{\mathsf{t}} + \mathsf{d}$
      else
        do nothing
      end if
    else if $\mathsf{r} \in \mathfrak{s}_{\mathsf{i}}^{\mathsf{t}}[1..\mathsf{R}]$ & $\mathsf{r} > \mathsf{R}$ **then** ▷ Agent_{r} *is not yet ranked in top* R
      buy $\mathsf{a} \cdot \mathsf{b}_{\mathsf{r},\mathsf{i}}^{\mathsf{t}}$ where $\mathsf{a} \overset{\mathsf{iid}}{\sim} \mathfrak{p}[0,1]$
    else
      do nothing
    end if
  end for
end while

Some commentary on AuditStrat follows:

1. The Auditors are prompted with a likely outcome $\mathfrak{s}_{\mathsf{r}}^{*}$, computed from the ranking over K models on the validation set. Critically, each $\mathsf{Auditor}_{\mathsf{i}}$ may accept this suggestion or use their own private belief $\mathfrak{q}_{\mathsf{i}}^{\mathsf{t}}$.

2. The $\mathsf{Auditor}_{\mathsf{i}}$ checks the current market price vector $\boldsymbol{\rho}_{\tau}^{\mathsf{t}}$, and determine its induces ordering $\mathfrak{s}_{\tau}^{\mathsf{t}}$. If the Auditor believes $\mathsf{Agent}_{\mathsf{r}}$'s model should be ranked within R, and his model is already ranked, then the Auditor flips a coin and either buys shares in $\mathsf{Auditor}_{\mathsf{i}}$ or do nothing. In the case where he buys shares, the $\mathsf{Auditor}_{\mathsf{i}}$ may buy shares in excess of what he would normally buy under expression 2.2 using private parameters C and d. This would occur if the $\mathsf{Auditor}_{\mathsf{i}}$ believes the respective position to r in stage $\mathsf{t}+1$ will be large, and he bids to claim a larger share of this pot. In other words, the $\mathsf{Auditor}_{\mathsf{i}}$ is in *competitive mode*.

3. In the event where $\mathsf{Agent}_{\mathsf{r}}$'s model is not yet ranked one of the top 1..R under the market's price vector's induced permutation, then the $\mathsf{Auditor}_{\mathsf{i}}$ is in risk-averse *cooperation* mode. In this case, he pursues a *mixed strategy* by sampling some constant $0 \leq \mathsf{a} \leq 1$ from distribution $\mathfrak{p}[0,1]$, and multiplies it with the constant $\mathsf{b}_{\mathsf{r},\mathsf{i}}^{\mathsf{t}} \geq 0$ as determined by 2.2. The intuition here is that $\mathsf{Auditor}_{\mathsf{i}}$ will signal quality to the market by buying some shares, but not so much that he is "sticking his lonesome neck out" for an $\mathsf{Agent}_{\mathsf{r}}$ whose quality may appear dubious to other Auditors.

4. If the auditor believes $\mathsf{Agent}_{\mathsf{r}}$ should not be top ranked at all, then he does nothing. In particular, notice the Auditor does not update his private belief $\mathfrak{q}_{\mathsf{i}}^{\mathsf{t}}$ as the $\mathsf{QAudit}_{\mathfrak{v}}$ market progresses.

Now observe as every $\mathsf{Auditor}_{\mathsf{i}}$ runs strategy $\mathsf{AuditStrat}$, the market inventory $\mathbf{s}^{\mathsf{t}}_{\tau}$ updates as they buy shares with $\left[\mathbf{b}^{\mathsf{t}}_{\mathsf{i},\tau_1}, \mathbf{b}^{\mathsf{t}}_{\mathsf{j},\tau_2}, \ldots\right]$. The corresponding market price vector $\boldsymbol{\rho}^{\mathsf{t}}_{\tau}$ changes, as does the induced ordering $\mathfrak{s}^{\mathsf{t}}_{\tau}$. So there is a one-to-one "*correspondence*" indexed by τ:

$$\underbrace{\left[\mathbf{s}^{\mathsf{t}}_{\tau_1}, \mathbf{s}^{\mathsf{t}}_{\tau_2}, \ldots, \mathbf{s}^{\mathsf{t}}_{\tau_{\mathsf{f}}}\right]}_{\text{share vectors}} \implies \underbrace{\left[\boldsymbol{\rho}^{\mathsf{t}}_{\tau_1}, \boldsymbol{\rho}^{\mathsf{t}}_{\tau_2}, \ldots, \boldsymbol{\rho}^{\mathsf{t}}_{\tau_{\mathsf{f}}}\right]}_{\text{price vectors}} \implies \underbrace{\left[\mathfrak{s}^{\mathsf{t}}_{\tau_1}, \mathfrak{s}^{\mathsf{t}}_{\tau_2}, \ldots, \mathfrak{s}^{\mathsf{t}}_{\tau_{\mathsf{f}}}\right]}_{\text{permutations}}.$$

The permutation states $\mathfrak{s}^{\mathsf{t}}_{\tau}$ express a *permutation state machine*, whereby the states are permutations, and the state transitions are permutation functions. Moreover, the updates in price space corresponds to a random walk on the permutation graph expressed by a PSA.

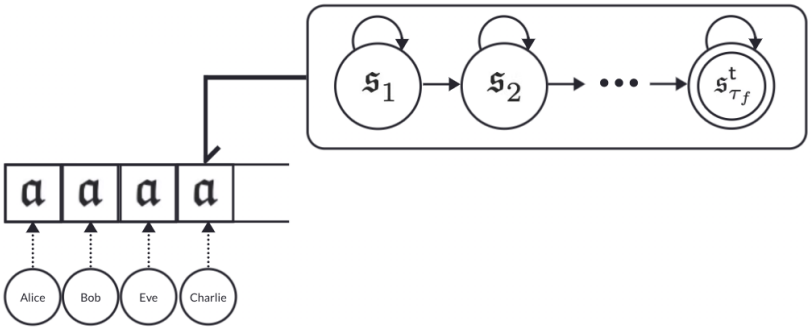


Figure 2.13: The QAudit market can be conceptualized as a permutation state machine (PSA). Each state $\mathfrak{s}_{\tau}, \ldots$ in the machine corresponds to a particular ranking of the Agents, and the edges are $\mathfrak{a} : \mathfrak{s}_{\tau} \to \mathfrak{s}_{\tau+1}$, which permutes the ranking over Agents. The PSA reads from a tape with symbols $\mathfrak{a}$, Auditors such as alice and bob place the $\mathfrak{a}$'s on the tape as they wager in the QAudit market. At market close, the PSA is in some state $\mathfrak{s}^{\mathsf{t}}_{\tau_{\mathsf{f}}}$, the top R Agents in this permutation wins round t. Thus we can conceptualize QAudit as a randomized algorithm as defined in B.1.1, so that $\mathsf{QAudit} : \mathbb{S}_{\mathsf{K}} \to \Delta(\mathbb{S}_{\mathsf{K}})$ accepts a suggested permutation $\mathfrak{s}^*_{\mathsf{r}}$, and output a distribution over the set of permutation $\mathbb{S}_{\mathsf{K}}$. Its source of randomness is the tape filled with $\mathfrak{a}$'s written by the Auditors. QAudit's payout mechanisms are designed so that in equilibrium, the likelihood that $\mathfrak{s}^{\mathsf{t}}_{\tau_{\mathsf{f}}} = \mathfrak{s}^*_{\mathsf{r}}$ is high.

Definition 2.5.6. *Permutation State Automata (*PSA*) with finite states*, or a permutation state machine expressing permutations over K integers is a four-tuple:

$$\mathsf{PSA} = \left(\mathbb{S}_{\mathsf{K}}, \mathbb{A}_{\mathsf{K}}, \mathfrak{s}_o, \mathbb{S}^*_{\mathsf{K}}\right),$$

so that:

2.5.6.1. the states are permutations, where every $\mathfrak{s}_{\mathsf{r}} \in \mathbb{S}_{\mathsf{K}}$ is some permutation of values 1...K, so that $|\mathbb{S}_{\mathsf{K}}| = \mathsf{K}!$.

2.5.6.2. The edge set $\mathbb{A}_{\mathsf{K}}$, where every edge $\mathfrak{a} := (\mathfrak{s}_{\mathsf{r}}, \mathfrak{s}_{\mathsf{s}}) \in \mathbb{A}_{\mathsf{K}}$ is defined as:

$$\mathfrak{a} : \mathbb{S}_{\mathsf{K}} \to \mathbb{S}_{\mathsf{K}}, \quad \mathfrak{a}(\mathfrak{s}_{\mathsf{r}}) = \mathfrak{s}_{\mathsf{s}},$$

that permutes some $\mathfrak{s}_{\mathsf{r}} \in \mathbb{S}_{\mathsf{K}}$ to create $\mathfrak{s}_{\mathsf{s}} \in \mathbb{S}_{\mathsf{K}}$. Observe given K integers, there are K! possible permutation actions, and at most $\frac{\mathsf{K}(\mathsf{K}-1)}{2}$ state transitions, including the self loop. We denote this self loop, or the identity permutation action with $\mathfrak{a}_{\mathsf{id}}$.

2.5.6.3. Initial state $\mathfrak{s}_o \in \mathbb{S}_{\mathsf{K}}$ that represent the initial permutation.

2.5.6.4. Set of terminal states $\mathbb{S}^*_{\mathsf{K}} \subseteq \mathbb{S}_{\mathsf{K}}$ that is "accepted" under some external criteria.

Note that the state and transitions of the PSA form a graph $\mathsf{G}\big(\mathbb{S}_{\mathsf{K}}, \mathbb{A}_{\mathsf{K}}\big)$, so normal graph concepts apply here. Finally, the PSA reads an external *state transition tape* whose symbol set is composed of transition actions $\mathfrak{a} \in \mathbb{A}_{\mathsf{K}}$, and undertakes the transition according to the $\mathfrak{a}$ on this tape. The frequency of said symbols that appear on the tape will correspond to weights or transition probability of the PSA, describing a weighted graph $\mathsf{G}\big(\mathbb{S}_{\mathsf{K}}, \mathbb{A}_{\mathsf{K}}, \mathbf{M}\big)$, with $\mathbf{M}$ as the transition probabilities.

Example 2.5.7. (PSA *with two states*). A PSA representing permutations over two integers is written:

$$\mathsf{PSA} = \Big(\mathbb{S}_{\mathsf{K}} = \big\{\mathfrak{s}_1, \mathfrak{s}_2\big\} = \big\{\big(1,2\big), \big(2,1\big)\big\}, \mathbb{A}_{\mathsf{K}} = \big\{(\mathfrak{s}_{\mathsf{i}}, \mathfrak{s}_{\mathsf{j}}) \,:\, \mathsf{i},\mathsf{j} \in \big\{1,2\big\}\big\}, \mathfrak{s}_o = \big(1,2\big), \mathbb{S}^*_{\mathsf{K}} = \big\{\big(2,1\big)\big\}\Big).$$

A *"correspondence"* between wager vector $\mathbf{b}^{\mathsf{t}}_{i\tau}$ and some transition matrix $\mathfrak{a}$ is the transition induced by this wager. For example, given initial market inventory $\mathbf{s}^{\mathsf{t}}_{\tau_{\mathsf{o}}} = \big(10, 5\big)$, with implied permutation $\mathfrak{s}^{\mathsf{t}}_{\tau_{\mathsf{o}}} = \big(1,2\big)$. If $\mathsf{Auditor}_{\mathsf{i}}$ places the buy vector $\mathbf{b}^{\mathsf{t}}_{\mathsf{i},\tau_{\mathsf{o}}} = \big(0, 7\big)$. Then the market inventory state is updated to $\mathbf{s}^{\mathsf{t}}_{\tau_1} = \big(10, 12\big)$, with associated updated price or probability vector $\boldsymbol{\rho}^{\mathsf{t}}_{\tau_1} = \big(0.117, 0.881\big)$, and implied ordering $\mathfrak{s}^{\mathsf{t}}_{\tau_1} = \big(2,1\big)$. Then $\mathbf{b}^{\mathsf{t}}_{\mathsf{i},\tau_{\mathsf{o}}}$ induces a symbol $\mathfrak{a}$ on the *transition tape* so that $\mathfrak{a}(\mathfrak{s}_1) = \mathfrak{s}_2$. Note depending on the size of the purchase, a wager vector $\mathbf{b}^{\mathsf{t}}_{\mathsf{i},\tau_{\mathsf{o}}}$ may not change the current market permutation at all, i.e the purchase vector $\mathbf{b}^{\mathsf{t}}_{\mathsf{i},\tau_{\mathsf{o}}} = \big(0, 4\big)$ corresponds to the identity permutation $\mathfrak{a}_{\mathsf{id}}$.

The PSA abstraction is useful in two ways:

1. PSA separates the permutation graph from the weight of the edges, which is determined by the distribution over symbols $\mathfrak{a}$ written onto the tape. Normally, one would place them together in a weighted permutation graph with $\mathsf{G}\big(\mathbb{S}_{\mathsf{K}}, \mathbb{A}_{\mathsf{K}}, \mathbf{M}\big)$, where $\mathbf{M}$ is the transition probabilities, or the "walk matrix." Separating the transition probability from the graph makes it clear that it is a property of the wagering behavior recorded on the tape, not an intrinsic feature of the permutation graph itself. Now we can ask questions such as how changes in audit strategy changes the transition probability, and subsequently the final outcome of $\mathsf{QAudit}_{\mathfrak{v}}$. From a computational standpoint, PSA is the same as a weighted graph, so all the classical graph concepts still apply.

2. Alternatively, if we conceptualize $\mathsf{QAudit}_{\mathfrak{v}}$ as a randomized algorithm, then the permutation matrices on the tape are the random bits, and the Auditors are the source of this randomness. We are interested in determining the likelihood that this randomized algorithm will terminate with the correct answer. In other words, if wagering actions induce a random walk over the PSA permutation graph, then we must show that it converges in reasonable number of wagers.

Notably, on the PSA tape, the distribution over permutation actions $\mathbb{A}_{\mathsf{K}}$ is determined by the Auditors' private belief $\mathfrak{s}^*_{\mathsf{r}}$, the constant C, d, and distribution over a in line 14 of AuditStrat. However, we do not express an explicit formula for the distribution as a function of C, d and a. Instead, we speak generally about convergence of the random walk governed by these constants, see B.1.4 and B.1.5 for relevant definitions.

Lemma 2.5.8. *In the event where every* $\mathsf{Auditor}_i$ *accepts the prompted ranking, and hold the corresponding private belief* $\mathfrak{s}^*_r$ *over a ranking of* K *models from* $\mathsf{Agent}_1, \ldots, \mathsf{Agent}_{\mathsf{K}}$. *Then the* Auditors *will collectively find the correct permutation in* $\mathsf{poly}\big(\mathsf{K!}\big)$ *number of wagers. In particular, we have the following bound on hitting time* $\mathsf{h}_{1,\mathsf{K!}}$*:*

$$\big(\mathsf{p}_a^{-1} + \mathsf{p}_a^0 + \mathsf{p}_a^1 + \mathsf{p}_a^2 + \ldots + \mathsf{p}_a^{\mathsf{K!}-2} + \mathsf{p}_a^{\mathsf{K!}-1}\big) \cdot \mathsf{K!} \leq \mathsf{h}_{1,\mathsf{K!}} \leq \frac{\sum_{k=0}^{\mathsf{K!}} \mathsf{p}_a^k}{\mathsf{p}_a \cdot \prod_{k=1}^{\mathsf{K!}-2}(1 - \mathsf{p}_{k,k})} \cdot \mathsf{K!}. \quad (2.4)$$

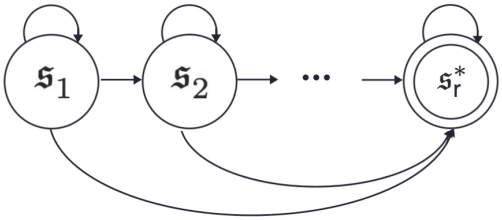


Figure 2.14: When every Auditor accept the prompted permutation $\mathfrak{s}_\mathsf{r}^*$, the directed permutation graph has a unique sink in $\mathfrak{s}_\mathsf{r}^*$. So every permutation action $\mathfrak{a}$ takes the PSA closer to this sink. Its hitting time is expressed in eqn 2.4.

Proof. See Appendix C.3 ■

Remark 2.5.9. (*Realistic wagering behavior*) Even though the Auditors converges onto the correct ordering in $\mathsf{poly}(\mathsf{K!})$ number of steps, the K! value is still large. However in reality the LMSR of eqn (2.2) increases the price on each buy signal, as it is proportional to the number of shares outstanding for every $\mathsf{Agent}_\mathsf{r}$ via: $\rho_\mathsf{r}^\mathsf{t} \sim \exp(\mathsf{s}_\mathsf{r}^\mathsf{t})$. The Auditor will weigh this increase in price on a per wager basis, and be less inclined to conservatively increment $\mathsf{Agent}_\mathsf{r}$'s position one position at a time. So in reality Auditors will not be crossing the entire span of the PSA graph in K! steps, but make large jumps as his risk profile sees fit.

In the lemma above we assume all the Auditors accept the QAudit prompt and thus share the same private belief over the ranking of players. This is a fair assumption as QAudit publishes the log-likelihood of data for every model $\boldsymbol{\theta}_1^\mathsf{t}, .., \boldsymbol{\theta}_\mathsf{K}^\mathsf{t}$ on every stage t, which is a good proxy for final ranking on the test set. The next lemma determines what is likely to occur if the Auditors do not share the same underlying belief. In this case the PSA expresses an undirected graph that does not have a unique sink. Therefore computing hitting time onto any state is vacuous, as the Auditors could always wager to exit this state. Instead we determine the steady state distribution over all states in the PSA graph, and express its gap w.r.t the solution found in the directed PSA. A few more definitions are needed before proceeding with the argument, in particular the idea of representing random walk in matrix form. They are found in B.1.6, B.1.7, and B.1.8.

In the case of the PSA, the probability of each permutation denotes the likelihood that Auditors will converge onto this solution under QAudit wagering. Theorem B.1.8 in the Appendix states that such a steady state must exist and can be found by solving eqn (B.1). Now we reference lemma 2.5.8, and let $\mathbf{M}_\mathsf{r}$ be the directed graph's walk matrix. Then following [SW22], we can construct the walk matrix $\mathbf{M}$ of the directed graph of 2.14 with a slight perturbation:

$$\mathbf{M} = (1-\epsilon)\mathbf{M}_\mathsf{r} + \epsilon\frac{\mathbf{e}\mathbf{e}^\intercal}{\mathsf{K!}}, \quad \text{where } \epsilon \sim \frac{1}{\mathsf{K!}}, \quad \text{and } \tilde{\boldsymbol{\delta}}_\mathsf{r} = \mathbf{p}\mathbf{M} \quad \text{for every } \mathbf{p} \in \mathfrak{P}. \tag{2.5}$$

Here $\mathbf{e}$ is the all one's vector of dimension $\mathsf{K!} \times 1$; and the uniform random walk matrix $\epsilon\frac{\mathbf{e}\mathbf{e}^\intercal}{\mathsf{K!}}$ has dimension $\mathsf{K!} \times \mathsf{K!}$. [10] This $\mathbf{M}$ is now invertible as there is a $\frac{1}{\mathsf{K!}^2}$ probability of a transition from any state onto another. The hitting time onto the sink associated with the PSA parameterized by $\mathbf{M}_\mathsf{r}$ is comparable to the mixing time of the undirected graph $\mathbf{M}$. Note we also have the stationary distribution of the PSA, which is approximately the "K-bit" vector $\tilde{\boldsymbol{\delta}}_\mathsf{r} = (\delta_1, \ldots, \delta_\mathsf{K})$, where $\delta_\mathsf{k} = 1$ if $\mathsf{k} = \mathsf{r}$, and $\delta_\mathsf{k} \sim 0$ otherwise. Critically, $\mathbf{M}$ is in fact the Google matrix, where:

1. the Markov transition matrix $\mathbf{M}_\mathsf{r}$ describes the internet links.

[10] Recall K is the number of Agents participating in an instance of QAudit.

2. $\frac{\mathbf{e}\mathbf{e}^\intercal}{\mathsf{K}!}$ is the "random surfer" behavior that renders $\mathbf{M}$ ergodic.

3. $\tilde{\boldsymbol{\delta}}_{\mathsf{r}}$ is PageRank, or a measure of network centrality of different vertices on the graph.

4. The most likely permutation here $\mathfrak{s}_{\mathsf{r}}^*$ is the top ranked website [BP98].

So we set out to determine how the PSA converges onto the best permutation state $\mathfrak{s}_{\mathsf{r}}^*$ in the setting where many Auditors holding different opinions, and reduced the problem to the sensitivity of network centrality measure on the PSA graph to changes in ϵ.

Lemma 2.5.10. *Suppose there is a block of* Auditors *who subscribe to the suggested* $\mathfrak{s}_r^*$ *on stage* t*, and other* Auditors *who hold* $m-1$ *(*$m-1<\mathsf{N}$*) different private beliefs. Let:*

2.5.10.1. *the private beliefs* $\{\mathfrak{q}_2,\ldots,\mathfrak{q}_m\}$ *induce a set of permutations* $\{\mathfrak{s}_2,\ldots,\mathfrak{s}_m\}$*;*

2.5.10.2. *the proportion of* Auditors *with each belief* $\mathfrak{s}_i$ *is* $\alpha_r,\ldots,\alpha_m$ *with* $\alpha_r+\sum_{i=2}^m\alpha_i=1$*;*

2.5.10.3. *and for now let* $\alpha_r\geq\alpha_2\geq\ldots\geq\alpha_m$*.*

If we separate each block of Auditors *onto their own* PSA *and use eqn 2.5, then they will induce stationary distributions* $\tilde{\boldsymbol{\delta}}_r,\ldots,\tilde{\boldsymbol{\delta}}_m$*. Now if the* Auditors *are playing at the same table as is the case in* QAudit*, and we take the perspective of the largest block of* Auditors *with belief* $\mathfrak{s}_r^*$*, then the difference between their optimal* $\tilde{\boldsymbol{\delta}}_r$ *and the stationary distribution is:*

$$\left\|\boldsymbol{p}-\tilde{\boldsymbol{\delta}}_r\right\|_1\leq\frac{2}{1-\alpha_r}|\Delta\alpha_r|,\ \text{ where }\boldsymbol{p}^\intercal=\boldsymbol{p}\mathbf{M},\ \ \mathbf{M}=\sum_i^m\alpha_i\mathbf{M}_i,\ \text{ and }\|\mathbf{x}\|_1=\sum_i|x_i|. \tag{2.6}$$

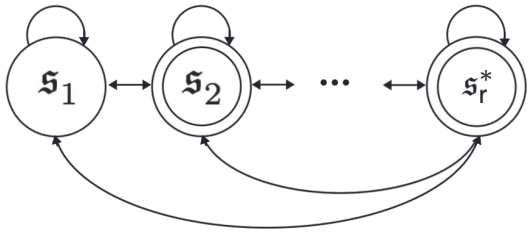


Figure 2.15: When Auditors have different opinions on what the best ranking over Agents are, the permutation graph does not have a unique sink. So we express a stationary distribution over all rankings in $\mathbb{S}_{\mathsf{K}}$ under the walk matrix $\mathbf{M}$ associated with this graph.

Proof. See Appendix C.3. ■

Next, we examine the sensitivity of $\tilde{\boldsymbol{\delta}}$ to changes in $\mathbf{e}$. That is to say instead of uniform preference over K Agents per $\mathbf{e}\mathbf{e}^\intercal/\mathsf{K}!$, suppose we know the Auditors who do not believe in $\mathfrak{s}_{\mathsf{r}}^*$ collectively have some biased preference $\mathbf{e}+\Delta\mathbf{e}$ over the Agents. Then we can build a teleportation matrix with this nonuniform distribution, and by similar reasoning to 2.5.10 find:

$$\|\tilde{\boldsymbol{\delta}}-\mathbf{p}\|_1\leq(1-\alpha_{\mathsf{r}})\|\Delta\mathbf{e}\|_1,\ \text{ where }\mathbf{p}^\intercal=\mathbf{p}\Big(\alpha_{\mathsf{r}}\mathbf{M}_{\mathsf{r}}+(1-\alpha_{\mathsf{r}})\mathbf{e}(\mathbf{e}+\Delta\mathbf{e})^\intercal\Big). \tag{2.7}$$

In the context of PageRank, this vector $\mathbf{e}+\Delta\mathbf{e}$ is the personalization vector. In our context, $\mathbf{e}+\Delta\mathbf{e}$ could be the initial distribution or price vector $\boldsymbol{\rho}^{\mathsf{t}}_{\tau_o}$ computed by $\mathsf{QAudit}_{\boldsymbol{\rho}^o}$, which describes how much each model $\mathbf{w}_{\mathsf{k}}^{\mathsf{t}}$ updated from the last stage's model $\mathbf{w}_{\mathsf{k}}^{\mathsf{t}-1}$. And, finally we can perturb the value of $\mathbf{M}_{\mathsf{r}}$ itself with $\Delta\mathbf{M}_{\mathsf{r}}$, and find the bound:

$$\|\tilde{\boldsymbol{\delta}}-\mathbf{p}\|_1\leq\frac{\alpha_{\mathsf{r}}}{1-\alpha_{\mathsf{r}}}\|\Delta\mathbf{M}_{\mathsf{r}}\|_\infty,\ \text{ where }\mathbf{p}^\intercal=\mathbf{p}\Big(\alpha_{\mathsf{r}}(\mathbf{M}_{\mathsf{r}}+\Delta\mathbf{M}_{\mathsf{r}})+(1-\alpha_{\mathsf{r}})\frac{\mathbf{e}\mathbf{e}^\intercal}{\mathsf{K}!}\Big), \tag{2.8}$$

as seen in [IW06, p. 3]. This perturbation of $\mathbf{M}_{\mathsf{r}}$ correspond to how risk-averse or aggressive the Auditors wager under AuditStrat. Equations 2.7 and 2.8 are the price of anarchy bounds of QAudit. They describe how "bad" does the QAudit solution degrade if the Auditors do not behave.

Now note although lemma 2.5.10 and subsequent bounds convey the l_1 distance between the distribution under $\mathbf{M}_{\mathsf{r}}$ (with belief $\mathfrak{s}_{\mathsf{r}}^*$) and the mixed distributions $\mathbf{M} = \sum_{\mathsf{k}}^{\mathsf{m}} \alpha_{\mathsf{i}} \mathbf{M}_{\mathsf{i}}$, under m blocks of Auditors. However it does not convey the relative network centrality of permutation $\mathfrak{s}_{\mathsf{r}}^*$ v.s. the other possible rankings. The next lemma considers a very specific case, and examine what will occur if some majority of the Auditors behave.

Lemma 2.5.11. *In the event where the* Auditor*s are partitioned into two blocks, so that some of the* Auditor*s accept the prompted permutation, denoted $\mathfrak{s}_r^*$, and some of which hold a different opinion $\mathfrak{s}_s$. If the proportion of* Auditor*s who hold private belief $\mathfrak{s}_r^*$ is greater than* 50%*, then the likelihood of finding the state $\mathfrak{s}_r^*$ is greater than landing in some no-man's land state that is neither $\mathfrak{s}_r^*$ nor $\mathfrak{s}_s$. Moreover, under some very mild conditions, the likelihood of finding one of the two accepted solution states may be significantly greater than $\frac{1}{2}$.*

Proof. See Appendix C.3. ■

Remark 2.5.12. (*Mixing Time for Random Walk on Symmetric Groups*) The PSA construction uses the term "permutation graph" loosely: although the states are labeled with permutation on K integers, this detail does not influence the analysis of the problem. However it is well known that permutations on sets form a symmetric group S_{K} with K! elements. We can use primitive permutations to generate the group, then study i.e. the mixing time of random walks on this S_{K}. In particular, the *transposition random walk* describes a process whereby a card is selected from the proverbial deck, and its position is swapped with another card in the same deck. This corresponds to the "slow walk" of lemma 2.5.8. It is known that the mixing time for the transposition random walk on S_{K} is $\mathsf{K} \log \mathsf{K}$ to reach uniform randomness, see [Mat20], [Lal18], and [Tan23], and [Nic19] for further discussion.

All in all, we began the subsection with the intuition that Auditors operate in a complex wagering environment. Whereby they would bet in a risk averse manner, relying on the action of other Auditors to move their preferred $\mathsf{Agent}_{\mathsf{r}}$ to the front of the pack. Then they would switch to a competitive mode and "double-down" on the winning $\mathsf{Agent}_{\mathsf{r}}$. The above three lemmas shows that under this mixed collaboration-to-competition regime, the Auditors can nonetheless:

1. "hit" the best permutation $\mathfrak{s}_{\mathsf{r}}^*$ in $\mathsf{poly}(\mathsf{K}!)$ steps - if every Auditor accepts the recommended ranking $\mathfrak{s}_{\mathsf{r}}^*$.
2. If some Auditors reject $\mathfrak{s}_{\mathsf{r}}^*$, then $\mathsf{QAudit}_{\mathfrak{v}}$ settles on a steady state distribution over $\mathbb{S}_{\mathsf{K}}$ that is of order $|\Delta\alpha_{\mathsf{r}}|$ away from the (near) delta distribution centered in $\mathfrak{s}_{\mathsf{r}}^*$, in $\mathsf{K} \log \mathsf{K}$ number of wagers.
3. If more than $\frac{1}{2}$ of the Auditors wager according to prompted $\mathfrak{s}_{\mathsf{r}}^*$, then the likelihood of hitting $\mathfrak{s}_{\mathsf{r}}^*$ can be at least $\frac{1}{2}$, if not significantly higher.

Finally, note that even if the Auditors who wager on $\mathfrak{s}_{\mathsf{s}} \neq \mathfrak{s}_{\mathsf{r}}^*$ wins stage t, no Auditor on stage t profits *if* the top models in $\mathfrak{s}_{\mathsf{s}}$ do not out-perform the previous ensemble model at stage $\mathsf{t}-1$. The existence of a hard objective metric during the reward phase prevents Auditors from benefiting by "mooning" a particular model, and using its absolute price as the sole criteria for eventual profit.

2.6 Fundamental Theorems of QAudit

We conclude the discussion on the Auditor-Auditor behavior by lifting concepts of the last section into the terminology of games, so as to:

1. state the existence of a *Correlated Equilibrium* in QAudit for every stage t;
2. determine under what conditions should the Auditors change their mind to conform to prevailing market trends.

The correlation setting models strategic interactions of rational agents complemented by a correlation device, that is a mediator who can recommend behavior but cannot enforce it [[Rou14], [FBS20]]. In the classical setting, the mediator exists because the players cannot coordinate to play the best set of actions due to the limitations of their selfish incentives. In the QAudit case, the correct permutation is difficult to identify without side information as the search space is of size $\mathsf{K}!$. Thus QAudit recommends the most likely permutation $\mathfrak{s}_{\mathsf{r}}^*$ over the K Agents. The Auditors can then wager according to this public ranking.

Fundamental Theorems within Stage t

Definition 2.6.1. (Auditor *Individual Action Space*) Under AuditStrat, Auditors purchase model shares with a sequence of buy vectors via $[\mathbf{b}^{\mathsf{t}}_{\mathsf{i},\tau_1}, \mathbf{b}^{\mathsf{t}}_{\mathsf{i},\tau_2}, \ldots]$. Since the PSA interfaces with the Auditors through the tape, let the induced sequence of permutation functions $\mathsf{qauditSeq}_{\mathsf{i}} = [\mathfrak{a}^{\mathsf{t}}_{\mathsf{i},\tau_1}, \mathfrak{a}^{\mathsf{t}}_{\mathsf{i},\tau_2}, \ldots, \mathfrak{a}^{\mathsf{t}}_{\mathsf{i},\tau_{\mathsf{f}}}]$ written onto this tape be their actions. That is to say the *action space* is:

$$\mathbb{A}_{\mathsf{K}}^{\tau_{\mathsf{f}}} = \bigtimes_{\tau_{\mathsf{j}}}^{\tau_{\mathsf{f}}} (\mathbb{A}_{\mathsf{K}})_{\tau_{\mathsf{j}}},$$

the τ_{f}-fold Cartesian product of permutation functions on K integers representing the K Agents. If the Auditor does not play at some time increment τ, it is convenient to say that he places the identity permutation $\mathfrak{a}_{\mathsf{id}}$ onto the tape instead. Next, define a distribution $\mathfrak{Q}_{\mathsf{i}}$ with support on $\mathbb{A}_{\mathsf{K}}^{\tau_{\mathsf{f}}}$, so that some $\mathsf{Auditor}_{\mathsf{i}}$ may sample a sequence of actions from $\mathbb{A}_{\mathsf{K}}^{\tau_{\mathsf{f}}}$ according to this $\mathfrak{Q}_{\mathsf{i}}$, written:

$$\mathsf{qauditSeq}_{\mathsf{i}} \overset{\mathsf{iid}}{\sim} \mathfrak{Q}_{\mathsf{i}}, \quad \mathsf{qauditSeq}_{\mathsf{i}} \in \mathbb{A}_{\mathsf{K}}^{\tau_{\mathsf{f}}}, \qquad \text{where } \mathsf{qauditSeq}_{\mathsf{i}} := \left[\mathfrak{a}^{\mathsf{t}}_{\mathsf{i},\tau_1}, \ldots, \mathfrak{a}^{\mathsf{t}}_{\mathsf{i},\tau_{\mathsf{f}}}\right].$$

And observe we use the notation $\mathsf{qauditSeq}_{\mathsf{i}}$ to denote a sequence of permutation functions sampled by $\mathsf{Auditor}_{\mathsf{i}}$. If every $\mathsf{Auditor}_{\mathsf{i}}$ samples just so and write their permutation symbols onto the PSA tape, then clearly we have $\tau_{\mathsf{f}} \times \mathsf{N}$ symbols on the tape over N Auditors at market close. Denote this with:

$$\mathsf{qauditSeq}_{1:\mathsf{N}} \overset{\mathsf{iid}}{\sim} \mathfrak{Q}, \quad \text{where } \mathsf{qauditSeq}_{1:\mathsf{N}} := \left[\mathfrak{a}^{\mathsf{t}}_{1,\tau_1}, \ldots, \mathfrak{a}^{\mathsf{t}}_{1,\tau_{\mathsf{f}}}, \ldots, \mathfrak{a}^{\mathsf{t}}_{\mathsf{N},\tau_1}, \ldots, \mathfrak{a}^{\mathsf{t}}_{\mathsf{N},\tau_{\mathsf{f}}}\right].$$

And similarly we can write $\mathsf{qauditSeq}_{1:\mathsf{N},-\mathsf{i}}$ as the sequence modulo the symbols from $\mathsf{Auditor}_{\mathsf{i}}$, this is every player's strategy except $\mathsf{Auditor}_{\mathsf{i}}$, so that:

$$\mathsf{qauditSeq}_{1:\mathsf{N}} := \mathsf{qauditSeq}_{\mathsf{i}} : \mathsf{qauditSeq}_{1:\mathsf{N},-\mathsf{i}}, \quad \text{where } \mathsf{qauditSeq}_{1:\mathsf{N}} \in (\mathbb{A}_{\mathsf{K}}^{\tau_{\mathsf{f}}})^{\mathsf{N}} = \bigtimes_{\mathsf{i}}^{\mathsf{N}} (\mathbb{A}_{\mathsf{K}}^{\tau_{\mathsf{f}}})_{\mathsf{i}}.$$

This is one example sequence of the symbols on the tape, highlighting $\mathsf{Auditor}_{\mathsf{i}}$'s moves.

Definition 2.6.2. (Auditor*'s Collective Action Space*) There exists a natural relationship between $\mathsf{qauditSeq}_{1:\mathsf{N}}$ and the walk matrix: the joint distribution $\mathfrak{Q} = \bigtimes_{\mathsf{i}}^{\mathsf{N}} \mathfrak{Q}_{\mathsf{i}}$ over the set of symbols $(\mathbb{A}_{\mathsf{K}}^{\tau_{\mathsf{f}}})^{\mathsf{N}}$ on the tape induces a set of random processes over the PSA. This is expressed as a convex cone of walk matrices:

$$\mathfrak{M}_{\mathsf{K}} = \left\{ \mathbf{M} : \ \mathbf{M} = \sum_{\mathsf{i}}^{\mathsf{m}} \alpha_{\mathsf{i}} \mathbf{M}_{\mathsf{i}} \ \text{ where } m \geq 1, \ \sum_{\mathsf{i}}^{\mathsf{m}} \alpha_{\mathsf{i}} = 1, \ \mathbf{M}_{\mathsf{i}} \text{ row-stochastic} \right\}.$$

$\mathfrak{M}_{\mathsf{K}}$ is the space of all row-stochastic walk matrices of dimension $\mathsf{K} \times \mathsf{K}$ closed under convex combination. For example in lemma 2.5.10, we had $\mathbf{M}_{\mathsf{r}}, \mathbf{M} \in \mathfrak{M}_{\mathsf{K}}$. And similarly the uniform walk matrix $\mathbf{e}\mathbf{e}^{\intercal}/\mathsf{K}!$ is also in $\mathfrak{M}_{\mathsf{K}}$, corresponding to a set of Auditors who wagers randomly.

Note the mapping between $(\mathbb{A}_{\mathsf{K}}^{\tau_f})^{\mathsf{N}}$ and $\mathfrak{M}_{\mathsf{K}}$ is onto, though not necessarily one-to-one. So depending on the proof, it may be convenient to use some $\mathbf{M}_{\mathsf{r}} \in \mathfrak{M}_{\mathsf{K}}$ as the Auditors' "individual action in the context of the collective." This is a way of codifying "herd behavior" that simplifies some argument greatly. Whereas choosing $\mathfrak{a}$ is akin to investing in an ad-hoc manner, choosing $\mathbf{M}$ is akin to "buying into an entire future" - the very definition of a lemming. [11] Playing with $\mathfrak{M}_{\mathsf{K}}$ entails a hierarchical sampling process, whereby at any point in time τ within stage t, the Auditor can choose any one of $\mathbf{M}_{\mathsf{i}} \in \mathfrak{M}_{\mathsf{K}}$, and then choose some action $\mathfrak{a}$ based on $\mathbf{M}_{\mathsf{i}}$ and place the symbol on the PSA tape. Critically, the game state is no longer some permutation $\mathfrak{s}$, but rather probability distribution $\mathbf{p}$ over all $\mathsf{K}!$ permutations. One can then step through the game states by multiplying any state by $\mathbf{M}_{\mathsf{r}}$ via $\mathbf{p}_{\tau_1}{}^{\intercal} = \mathbf{p}_{\tau_0}\mathbf{M}_{\mathsf{r}}$, thus all prior arguments carry over.

Definition 2.6.3. (Auditor *Utility Function*) We can select any suitable permutation distance function denoted with $x \mid\mid y$, i.e. Kendall-Tau or Cayley distance. And define the $\mathsf{Auditor}_{\mathsf{i}}$'s utility function with:

$$\mathbf{u}_{\mathsf{i}} : \mathbb{A}_{\mathsf{K}}^{\tau_f} \times \mathbb{D} \to \mathbb{R} \qquad \mathbf{u}_{\mathsf{i}}\big(\mathsf{qauditSeq}_{i:\mathsf{N}}, \mathbb{D}_{\mathsf{val}}\big) = \mathbf{E}\left[\mathfrak{v}_{\mathsf{i}}^{\mathsf{t}} | \mathbb{D}_{\mathsf{val}}\right] \cdot \frac{1}{1 + (\mathfrak{s}_{\tau_f}^{\mathsf{t}} \mid\mid \mathfrak{s}_{\mathsf{i}}^{\mathsf{t}})},$$

where $\mathfrak{s}_{\mathsf{r}}^{*\mathsf{t}}$ is the $\mathsf{QAudit}_{\mathfrak{v}}$-suggested permutation computed over the validation set $\mathbb{D}_{\mathsf{val}}$ in stage t. Observe this utility function is a product for the *expected reward* $\mathfrak{v}_{\mathsf{i}}^{\mathsf{t}}$ metered out by $\mathsf{QAudit}_{\mathfrak{v}}$ under data distribution, and the permutation distance between the Auditor's ideal permutation $\mathfrak{s}_{\mathsf{i}}^{\mathsf{t}}$, and the actual permutation at market close $\mathfrak{s}_{\tau_f}^{\mathsf{t}}$. In particular, by maximizing $\mathbf{u}_{\mathsf{i}}$, $\mathsf{Auditor}_{\mathsf{i}}$ increases the probability that test-data likelihood under ensemble model in stage t outperforms that of $\mathsf{t}-1$, and thus can claim reward from wagers collected in stage $\mathsf{t}+1$.

Theorem 2.6.4. *(*QAudit *Protocol Consensus Theorem) For every wager stage* t*, there exists a Correlated Equilibrium (CE) (see definition B.2.3) such that the market consensus at market close reflects the true permutation* $\mathfrak{s}_{\mathsf{r}}^{*\mathsf{t}}$ *in expectation, provided the permutation is prompted by* $\mathsf{QAudit}_{\mathfrak{v}}$ *at market open in every* t.

Proof. Since this is an existence proof, we submit the randomized audit program $\mathsf{AuditStrat}(\mathfrak{s}_{\mathsf{r}}^{*}, C, d)$ to witness the existence of a viable strategy. Then if we denote $\mathsf{qauditSeq}_{\mathsf{i}}$ as a strategy for $\mathsf{Auditor}_{\mathsf{i}}$ if he accepts the protocol-prompted $\mathfrak{s}_{\mathsf{r}}^{*\mathsf{t}}$, and let $\mathsf{qauditSeq}_{\mathsf{i}}'$ be the wagering strategy if he rejects the prompt. Then for every $\mathsf{Auditor}_{\mathsf{i}}$, we can say the following:

$$\begin{aligned}\mathbf{E}_{\mathbb{D}\sim\mathfrak{D}}\left[\mathbf{E}_{\mathsf{qauditSeq}_{1:\mathsf{N}}\sim\mathfrak{Q}}\left[\mathbf{u}_{\mathsf{i}}\big(\mathsf{qauditSeq}_{\mathsf{i}} : \mathsf{qauditSeq}_{1:\mathsf{N},-\mathsf{i}}, \mathbb{D}\big)\middle|\mathfrak{s}_{\mathsf{r}}^{*\mathsf{t}}\right]\right] \geq \\ \mathbf{E}_{\mathsf{qauditSeq}_{1:\mathsf{N}}\sim\mathfrak{Q}}\left[\mathbf{u}_{\mathsf{i}}\big(\mathsf{qauditSeq}_{\mathsf{i}}' : \mathsf{qauditSeq}_{1:\mathsf{N},-\mathsf{i}}, \mathbb{D}\big)\middle|\mathfrak{s}_{\mathsf{r}}^{*\mathsf{t}}\right]\right].\end{aligned} \tag{2.9}$$

This is our form of the correlated equilibrium condition, whereby:

2.6.4.1. the outer expectation $\mathbf{E}_{\mathbb{D}\sim\mathfrak{D}}\left[\cdot\right]$ is taken w.r.t all samples of test/validation set $\mathbb{D}$ from data distribution $\mathfrak{D}$.

2.6.4.2. The statement before the inequality defines the expected utility value if the Auditor plays according to QAudit-suggested permutation $\mathfrak{s}_{\mathsf{r}}^{*\mathsf{t}}$.

2.6.4.3. The statement after the inequality defines the expected utility value if the $\mathsf{Auditor}_{\mathsf{i}}$ does not play according to $\mathfrak{s}_{\mathsf{r}}^{*\mathsf{t}}$.

2.6.4.4. both statements are conditioned on $\mathfrak{s}_{\mathsf{r}}^{*\mathsf{t}}$, the QAudit-suggested permutation.

This statement 2.9 follows from the lemmas of the previous section:

[11] So the euphemism for lemming is a random walker.

2.6.4.1. by $\mathsf{QAudit}_{\mathbb{D}}$ we know that the test set $\mathbb{D}_{\mathsf{test}}$ and validation set $\mathbb{D}_{\mathsf{val}}$ are drawn from the same distribution, so validation set permutation $\mathfrak{s}_{\mathsf{r}}^{*\mathsf{t}}$ is a fair representation of the true permutation $\mathfrak{s}_{\mathsf{T}}^{*\mathsf{t}}$ under test set.

2.6.4.2. If every $\mathsf{Auditor}_{\mathsf{i}}$ plays according to $\mathfrak{s}_{\mathsf{r}}^{*\mathsf{t}}$, then by lemma 2.5.8 there is a random walk s.t. the Agents can arrive at the target in polynomial hitting time. By setting the wagering stage to be greater than the hitting time, the PSA will converge on the correct permutation state at market close when Auditors wager under AuditStrat.

2.6.4.3. In the event where some portion $(1-\alpha_{\mathsf{r}})$ of the Auditors reject the suggested permutation $\mathfrak{s}_{\mathsf{r}}^{*\mathsf{t}}$ and wager according to $(1-\alpha_{\mathsf{r}})(\mathbf{M}_2+\ldots+\mathbf{M}_{\mathsf{m}})$, then by lemma 2.5.10 the near delta distribution $\tilde{\boldsymbol{\delta}}$ centered at $\mathfrak{s}_{\mathsf{r}}^{*\mathsf{t}}$ under $\mathbf{M}_{\mathsf{r}}$ degrades according to $\frac{2}{1-\alpha_{\mathsf{r}}}|\Delta\alpha_{\mathsf{r}}|$.

2.6.4.4. In fact, regardless of how some $(1-\alpha_{\mathsf{r}})$ portion of Auditors who do not believe in $\mathfrak{s}_{\mathsf{r}}^{*\mathsf{t}}$ wager, eqn 2.7 shows that the gap between the full consensus case of 2.5.8 and the degraded case is of order $(1-\alpha_{\mathsf{r}})\|\Delta\mathbf{e}\|_1$. When α_{r} is large, this difference is small.

2.6.4.5. Specifically, by lemma 2.5.11, we know that as long as more than $\frac{1}{2}$ of the Auditors play according to $\mathsf{qauditSeq}_{1:\mathsf{N}}$, then the likelihood of finding $\mathfrak{s}_{\mathsf{r}}^{*\mathsf{t}}$ may be far greater than $\frac{1}{2}$, depending on how aggressively the Auditors wager.

The three quantities of $|\Delta\alpha_{\mathsf{r}}|$, $\|\Delta\mathbf{e}\|_1$, and $\|\Delta\mathbf{M}_{\mathsf{r}}\|_\infty$ measure the gap between the best possible outcome under data expectation, and the CE outcome. All in all, this proves the existence and stability of correlated equilibrium as defined. ■

Following an earlier comment, it is not a given that the existence of objective metric at reward time would necessarily deter "mooning." The reward phase takes place after the audit phase has ended, and the audit phase alone may last weeks. Concurrently, Auditors can sell their positions on $\mathsf{mint}_k^{\mathsf{t}}$ in the secondary market. Thus we still need the analysis of theorem 2.6.4 to show that QAudit consensus mechanism deters the Auditors from manipulating the market irresponsibly. The QAudit consensus theorem states that unless the Auditors in the primary market are 1) "very sure" some other permutation is the correct choice, *and/or* 2) can convince the secondary buyers that they are sure, then the Auditors are better off wagering according to the suggested ranking $\mathfrak{s}_{\mathsf{r}}^{*\mathsf{t}}$.

Remark 2.6.5. (*Proof Methodology*). In the Auditor-Auditor proofs, we assume the existence of some *Homo Economicus* who wagers according to rational self interest, not emotions or loyalty. Normally, analysis that conform to neo-classical orthodoxy is not to be taken seriously. However in the QAudit case there is some justification. It is well known that most trades in DeFi are done by automated algorithms. Thus AuditStrat is not a fictitious rational Auditor, but rather a very real auditing algorithm. In fact, part of Metheus's go-to-market strategy is attracting DeFi "whales" and connecting them with these auditing strategies. This allows us to quickly build market depth and influence market variance. This last point is critical: remember the winning Agents are paid by the losing Auditors.

The remark above underlines the fact that if the winning Agents are to be paid, then there must be variance in how Auditors wager. Variance may arise spontaneously as Auditors disagree, or it can be constructed by the protocol itself. For example, QAudit may publish many possible final rankings $\mathfrak{s}_1^*,..,\mathfrak{s}_{\mathsf{m}}^*$, done on m different sets of validation data of slightly different distributions. And then prompt the Auditors with this set, so as to seed different fractions and promote market diversity. Now in every analysis thus far, we assume every $\mathsf{Auditor}_{\mathsf{i}}$ choose to accept or reject the market-suggested ranking $\mathfrak{s}_{\mathsf{r}}^*$ at market open within every stage t. And then plays strategy $\mathsf{AuditStrat}(\mathfrak{s}_{\mathsf{r}}^*, C, d)$ without ever updating their private belief w.r.t market condition. In reality people do change their mind, and this can be to our benefit. The next theorem relaxes the non-updating prior constraint, so that the Auditors may update their private beliefs per market state. In this case, "momentum trading" helps the market to arrive at one of the possible permutation candidates, with $\mathsf{m}-1$ losers whose wagers will be distributed to the winning Agents.

Theorem 2.6.6. *(*QAudit *Social Consensus Theorem) Suppose* QAudit *seeds the market with candidate final rankings* $\mathfrak{s}_1^*, \dots, \mathfrak{s}_m^*$ *on the test set in every stage* t*, so that the* Auditor*s may play according to any one of the mixed strategies:*

$$\mathfrak{M}'_{\mathsf{K}} = \Big\{ \mathsf{AuditStrat}(\mathfrak{s}_1^*, C, d), \dots, \mathsf{AuditStrat}(\mathfrak{s}_m^*, C, d) \Big\} \implies \Big\{ \mathbf{M}_1, \dots, \mathbf{M}_m \Big\}.$$

And note that $\mathfrak{M}'_{\mathsf{K}} \subset \mathfrak{M}_{\mathsf{K}}$*, and every mixed strategy* $\mathsf{AuditStrat}(\mathfrak{s}_{\mathsf{s}}^*, C, d)$ *correspond to a unique walk matrix* $\mathbf{M}_s$*. Then in every stage* t*, there exists a point in time* $\tau^* \in \{\tau_1, \dots, \tau_{\mathsf{f}}\}$ *s.t. it is rational for every* Auditor *to switch to a single mixed strategy* $\mathsf{AuditStrat}(\mathfrak{s}_r^*, C, d)$*, with corresponding game dynamic* $\mathbf{M}_r$*, favoring one ranking* $\mathfrak{s}_r^*$ *over all others.*

Proof. We use the collective actions space $\mathfrak{M}_{\mathsf{K}}$ of definition 2.6.2. Now the QAudit game can be expanded as a game tree, where the inner/outer leaves are $\mathbf{p}$, or distribution over the set $\mathbb{S}_{\mathsf{K}}$ with $\mathsf{K}!$ possible rankings, and the edges are walk matrices $\mathbf{M}_{\mathsf{r}} \in \mathfrak{M}'_{\mathsf{K}}$. That is to say, at any vertex on the tree, the $\mathsf{Auditor}_{\mathsf{i}}$ steps to the next vertex using some mixed strategy according to $\mathsf{AuditStrat}(\mathfrak{s}_{\mathsf{s}}^*, C, d) \implies \mathbf{M}_s$, not a weighted sum as was the case in eqn C.4. See [Tad13, p. 91] for more on game trees. The tree is of height τ_{f}, and at every inner leaf there are at least m possible expansions or branches, corresponding to the $\geq \mathsf{m}$ walk matrices in $\mathfrak{M}_{\mathsf{K}}$. There are at least $\mathsf{m}^{\tau_{\mathsf{f}}}$ possible paths down the tree. Any path down the tree then is of form:

$$\mathbf{p}_{\tau_f}{}^{\mathsf{T}} = \mathbf{p}_{\tau_{\mathsf{o}}} \Big(\underbrace{\mathbf{M}_{1,\mathsf{s}} \circ \dots \circ \mathbf{M}_{\tau^*,\mathsf{j}}}_{\text{exploration}} \circ \underbrace{\mathbf{M}_{\tau^*+1,\mathsf{r}} \circ \dots \circ \mathbf{M}_{\tau_{\mathsf{f}},\mathsf{r}}}_{\text{exploitation}} \Big), \quad \text{where } \mathsf{s}, \mathsf{r}, \mathsf{j} \in \big\{ 1 \dots \mathsf{m} \big\}.$$

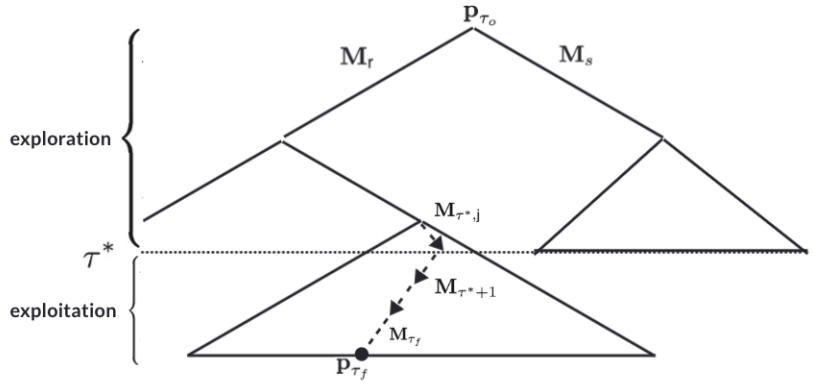


Figure 2.16: In this formulation of QAudit equilibrium condition, the Agents are prompted with many possible permutations and hold hetergeneous beliefs. They are collectively doing m-ary tree search to find the most profitable stationary distribution (binary tree search shown for clarity). Theorem 2.6.6 shows that rational Agents will *explore* any one of the suggested options, playing any $\mathsf{AuditStrat}\ \mathbf{M}_{\mathsf{r}}, \mathbf{M}_{\mathsf{s}} \in \mathfrak{M}_{\mathsf{K}}$. But beyond t^*, they will converge onto a consensus, and *exploit* it so that the stationary distribution at market close $\mathbf{p}_{\tau_{\mathsf{f}}}$ is centered on a unique ranking.

Now recall remark 2.5.12, stating the mixing time of a permutation random walk is of order $\mathsf{K}\log(\mathsf{K})$. So at some tree depth $\tau^* < \tau_{\mathsf{f}}$, the Auditors must make a decision: 1) wager with their existing mixed strategy $\mathsf{AuditStrat}(\mathfrak{s}_{\mathsf{s}}^*, C, d)$, or 2) follow the herd so as to maximize the likelihood that the random walk converges to any $\mathfrak{s}_{\mathsf{r}}^* \in \big\{ \mathfrak{s}_1^*, \dots, \mathfrak{s}_{\mathsf{m}}^* \big\}$. We can reason as follows:

2.6.6.1. If every Auditor stay with their existing strategy, then under the combined permutation matrix $\mathbf{M}^- = \alpha_1\mathbf{M}_1 + ... + \alpha_{\mathsf{m}}\mathbf{M}_{\mathsf{m}}$, the steady state distribution under $\mathbf{M}^-$ will be of order $2|\Delta\alpha_{\mathsf{r}}|$ away from the near-delta distribution $\tilde{\boldsymbol{\delta}}_{\mathsf{r}}$ for every $\mathsf{r} = 1...\mathsf{m}$ as seen in lemma 2.5.10. Moreover, by 2.5.11 we know that if no block of Auditors have a simple majority, then no permutation $\mathfrak{s}^*_{\mathsf{r}}$ may dominate the others in likelihood.

2.6.6.2. Similar to definition 2.6.3, we can formalize this idea by assigning a utility value $\mathbf{u}_{\mathsf{i}}$ to every terminal leaf in the space of terminal $\mathsf{m}^{\tau_{\mathsf{f}}}$ outcomes via:

$$\mathbf{u}_{\mathsf{i}} : \mathfrak{P} \to \mathbb{R}, \text{ where } \mathbf{u}_{\mathsf{i}}(\mathfrak{p}) = v,\ v \in \mathbb{R}, \text{ and } \mathfrak{P} = \big\{\mathfrak{p}(\mathfrak{s}) :\ \mathfrak{s} \in \mathbb{S}_{\mathsf{K}}\big\}.$$

Now suppose for some $\mathsf{Auditor}_{\mathsf{i}}$ ($\mathsf{i} = 1...\mathsf{N}$), we have an ordering over preferences s.t.:

$$\underbrace{\mathbf{u}_{\mathsf{i}}(\tilde{\boldsymbol{\delta}}^*_1) \geq \ldots \geq \mathbf{u}_{\mathsf{i}}(\tilde{\boldsymbol{\delta}}^*_{\mathsf{m}})}_{\text{preference over prompted permutations}} \quad > \quad \underbrace{\mathbf{u}_{\mathsf{i}}(\mathbf{p}_{\mathsf{m}+1}) \geq \ldots \geq \mathbf{u}_{\mathsf{i}}(\mathbf{p}_{\mathsf{m}^{\tau_{\mathsf{f}}}}),}_{\text{preference over unprompted permutations}}$$

$$\text{with } \mathbf{u}_{\mathsf{i}}(\tilde{\boldsymbol{\delta}}_{\mathsf{j}}) > 0, \text{ and } \frac{\mathsf{N}-1}{\mathsf{N}+1} < \frac{\mathbf{u}_{\mathsf{i}}(\tilde{\boldsymbol{\delta}}_{\mathsf{j}})}{\mathbf{u}_{\mathsf{i}}(\tilde{\boldsymbol{\delta}}_{\mathsf{l}})} < \frac{\mathsf{N}+1}{\mathsf{N}-1} \text{ for every } \mathsf{j}, \mathsf{l} \in 1...\mathsf{m}.$$

Here $\mathbf{p}_{\mathsf{m}+l}$ is the distribution, possibly uniform, over permutations not prompted by QAudit; while $\tilde{\boldsymbol{\delta}}^*_{\mathsf{j}}$ is the near-delta distribution centered on prompted permutation $\mathfrak{s}^*_{\mathsf{j}}$. Notably:

a) we assume the Auditor strictly prefers outcomes that favor the prompted permutations over the unprompted ones. That is $\mathbf{u}_{\mathsf{i}}(\tilde{\boldsymbol{\delta}}^*_{\mathsf{r}}) > \mathbf{u}_{\mathsf{i}}(\mathbf{p}_{\mathsf{j}})$ for every $\mathsf{r} = 1...\mathsf{m}$, and $\mathsf{j} = \mathsf{m}+1, \ldots, \mathsf{m}^{\tau_{\mathsf{f}}}$.

b) We place a constraint on the relative value of preferences over the prompted permutations. Thus modeling a set of Auditors who are not sure which of the suggested permutation is best - a reasonable assumption given 1) how close the validation sets are to each other, and 2) the assumed nature of DeFi users.

Now this $\mathsf{Auditor}_{\mathsf{i}}$ may *heuristically* assign the likelihood of achieving any $\tilde{\boldsymbol{\delta}}^*_{\mathsf{j}}$ by the proportion of Auditors who have wagered on it, namely α_{j}. His expected value for wagering on any outcome is then:

$$\mathbf{E}[\mathbf{u}_{\mathsf{i}}(\tilde{\boldsymbol{\delta}}_{\mathsf{j}})] = \alpha_{\mathsf{j}}\mathbf{u}_{\mathsf{i}}(\tilde{\boldsymbol{\delta}}_{\mathsf{j}}).$$

In fact, an expected value may be assigned to every one of the $\mathsf{m}^{\tau_{\mathsf{f}}}$ terminal leaves on the game tree. Now suppose this $\mathsf{Auditor}_{\mathsf{i}}$ is part of block α_{s} believing in $\mathfrak{s}^*_{\mathsf{s}}$, and he observes that there is some block α_{r} betting on $\mathfrak{s}^*_{\mathsf{r}}$, with $\alpha_{\mathsf{r}} > \alpha_{\mathsf{s}}$. Then he may defect if *after* he switches, he achieves $\mathbf{E}[\mathbf{u}_{\mathsf{i}}(\tilde{\boldsymbol{\delta}}_{\mathsf{r}})] > \mathbf{E}[\mathbf{u}_{\mathsf{i}}(\tilde{\boldsymbol{\delta}}_{s})]$. So his hurdle condition for defecting is:

$$\left(\alpha_{\mathsf{r}} + \frac{1}{\mathsf{N}}\right)\mathbf{u}_{\mathsf{i}}(\tilde{\boldsymbol{\delta}}_{\mathsf{r}}) > \left(\alpha_{\mathsf{s}} - \frac{1}{\mathsf{N}}\right)\mathbf{u}_{\mathsf{i}}(\tilde{\boldsymbol{\delta}}_{s}) \implies \frac{\alpha_{\mathsf{r}}\mathsf{N}+1}{\alpha_{\mathsf{s}}\mathsf{N}-1} > \frac{\mathbf{u}_{\mathsf{i}}(\tilde{\boldsymbol{\delta}}_{s})}{\mathbf{u}_{\mathsf{i}}(\tilde{\boldsymbol{\delta}}_{\mathsf{r}})}. \tag{2.10}$$

The last statement is true since $\frac{\alpha_{\mathsf{r}}\mathsf{N}+1}{\alpha_{\mathsf{s}}\mathsf{N}-1} > \frac{\mathsf{N}+1}{\mathsf{N}-1}$ for every $0 < \alpha_{\mathsf{r}}, \alpha_{\mathsf{s}} < 1$. Furthermore, note we have $\frac{\alpha_{\mathsf{r}}\mathsf{N}+1}{\alpha_{\mathsf{s}}\mathsf{N}-1} > \frac{\alpha_{\mathsf{j}}\mathsf{N}+1}{\alpha_{\mathsf{s}}\mathsf{N}-1}$ if $\alpha_{\mathsf{r}} > \alpha_{\mathsf{j}}$. Reasoning like so for every $\mathsf{j} \in \{1...\mathsf{m}\}/\{\mathsf{r}, \mathsf{s}\}$, this $\mathsf{Auditor}_{\mathsf{i}}$ must defect from his $\mathfrak{s}^*_{\mathsf{s}}$ towards the $\mathfrak{s}^*_{\mathsf{r}}$ with the highest α_{r}, and wager with $\mathbf{M}_{\mathsf{r}}$ from this point thence.

2.6.6.3. All in all, we see that for every time step after τ^*, there exists at least one $\mathsf{Auditor}_{\mathsf{i}}$ who switches to the highest α_{r}. Now by setting the threshold time with:

$$\tau_{\mathsf{f}} - \mathsf{N} - \mathsf{K}\log\big(\mathsf{K}\big) \leq \tau^* < \tau_{\mathsf{f}} - 1 - \mathsf{K}\log\big(\mathsf{K}\big),$$

then there is enough time for every $\mathsf{Auditor}_{\mathsf{i}}$ to switch to belief $\mathfrak{s}^*_{\mathsf{r}}$. And the chain still has time to mix and settle on $\tilde{\boldsymbol{\delta}}^*_{\mathsf{r}}$. That is to say after time step $\tau_{\mathsf{f}} - \mathsf{K}\log\big(\mathsf{K}\big)$, we can reuse the argument of eqn C.4 to construct the weighted walk matrix:

$$\mathbf{M}^+ = (1-\epsilon)\mathbf{M}_{\mathsf{r}} + \epsilon\frac{\mathbf{e}\mathbf{e}^{\intercal}}{\mathsf{K}!}, \text{ with } \epsilon \sim 0,$$

converging to $\tilde{\boldsymbol{\delta}}_{\mathsf{r}}^{*}$ in $\mathsf{K}\log(\mathsf{K})$ time. Furthermore, since this is a game of finite horizon, this sub-game equilibrium is also the game equilibrium.

■

Some final observations:

1. Defecting to the market simple majority at time τ^{*} is an example of a sub-game equilibrium [Tad13, p. 113]. In $\mathsf{QAudit}_{\mathfrak{v}}$ prompted by m permutations $\mathfrak{s}_{1}^{*}, .., \mathfrak{s}_{\mathsf{m}}^{*}$, with Auditors that can defect to any fraction, there are m sub-game equilibria.

2. In the event where for some $\mathsf{Auditor}_{\mathsf{i}}$ his preference is such that $\frac{\mathbf{u}_{\mathsf{i}}(\tilde{\boldsymbol{\delta}}_{s})}{\mathbf{u}_{\mathsf{i}}(\tilde{\boldsymbol{\delta}}_{\mathsf{r}})} > \frac{\mathsf{N}+1}{\mathsf{N}-1}$, then this $\mathsf{Auditor}_{\mathsf{i}}$ may not defect from his choice of $\mathfrak{s}_{\mathsf{s}}^{*}$ even in the face of another coalition. So the $\mathsf{Auditor}_{\mathsf{i}}$ has to make a decision, either:

   a) bet with conviction and purchase enough shares to move the market towards $\mathfrak{s}_{\mathsf{s}}^{*}$, and force other coalitions to respond. This outcome is OK from the perspective of QAudit, as long as $\mathfrak{s}_{\mathsf{s}}^{*}$ is one of the suggested permutations.

   b) Sit out the market for the remainder of stage t. We can encode his portion of the bets with the identity matrix $\mathbf{I}$, stating that he does not change the market state. If sufficient number of Agents sit out, then we have a walk matrix of form:

$$\mathbf{M} = (1 - \epsilon - \epsilon')\mathbf{M}_{\mathsf{r}} + \epsilon'\mathbf{I} + \epsilon\frac{\mathbf{e}\mathbf{e}^{\intercal}}{\mathsf{K}!},$$

   which also converges onto $\tilde{\boldsymbol{\delta}}_{\mathsf{r}}$ albeit with a slower mixing time due to $\epsilon'\mathbf{I}$.

3. The proof above assumes $\mathsf{N} + \mathsf{K}\log(\mathsf{K}) < \tau_{\mathsf{f}}$, in the event where τ_{f} is less than the hurdle, then the Auditors may not have enough time to defect and hit the final permutation $\mathfrak{s}_{\mathsf{r}}^{*}$. In reality, τ_{f} expresses the total number of trades within some stage t, so this implies the trading volume in any stage must be greater than $\mathsf{N} + \mathsf{K}\log(\mathsf{K})$.

4. All the Auditors' wagers before τ^{*} essentially does not matter, as the Markov chain can converge onto $\tilde{\boldsymbol{\delta}}_{\mathsf{r}}^{*}$ from any initial distribution $\mathbf{p}_{\tau^{*}}$ at time τ^{*}. This "burn in" period would ideally feature a diversity of wagers over m possible permutations, the losing wagers associated with the $\mathsf{m}-1$ losing permutations will accrue to the winning Agents at stage t. This benefit Agent retention.

5. Meanwhile, the competitive phase does not have to begin when some specific $\mathfrak{s}_{\mathsf{r}}^{*}$ is hit, but can instead begin at τ^{*} when Auditors "sense" the market is about to converge onto one final consensus. A long competitive phase leads to a larger wagering pool that is rewarded to winning Auditors in stage $\mathsf{t}-1$. This promotes Auditor retention.

This concludes the discussions on the fundamental theorems. However, it is important to state that we have just scratched the surface of what can be analyzed within the PSA computational context. In the last two sections, we provided one example of an audit strategy, and use it to argue that the PSA converges onto the best solution. In general, it is far more interesting to construct many variations of audit strategies, game them against each other on the PSA tape, and study the convergence behavior of the induced weighted PSA graph. This thought process will refine the consensus mechanism of the QAudit market maker.

Fundamental Theorem across Stages T

In the last few sections, we proved that:

1. Agent should report the best model they have on hand, subject to some resource considerations.

2. When Auditors act on a strictly individual basis, or the first Auditor who wagers in any given stage t, should wager on the best model submitted in that stage.

3. Auditors may alternate between competing to cooperating with each other. When deciding on what to wager w.r.t market consensus, theorems 2.6.4 and 2.6.6 states that it is in the Auditor's self interest to:

   a) wager according to the protocol's prompted permutation if one suggestion is given.

   b) If QAudit suggests more than one permutation, then Auditors should all defect to the market simple majority.

However, all the proofs are equilibrium statements within some stage t. This section extends the discussion to the game over all stages $\mathsf{t} = 1 \dots \mathsf{T}+1$, and argue that it is in the interest of the Auditors to continue to wager across all $\mathsf{T}+1$ stages. This is no mere academic point: this section is fundamentally about Auditor retention.

Now recall QAudit's staggered sequential parimutuel incentive structure as seen in table 2.5: it is a zero sum game until the last stage T when the bounty is paid out. That is to say one Auditor's gain on stage t is another Auditor's loss in stage $\mathsf{t}+1$. Clearly, no Auditor would wager in every stage, unless the snake eats its own tail. Instead they would naturally fracture into two cliques: call them Odds and Evens Auditors. The Odds wager on stages $\mathsf{t} = 1, 3, 5, \dots$, and the Evens wager on stages $\mathsf{t} = 2, 4, 6, \dots$. The immediate question then is: how stable is this game. In fact the problems are more numerous than this. Over horizon $\mathsf{T}+1$ the relevant problems are:

1. The tit-for-tat dynamic between the Odds and Evens cliques may be unstable. As long as both cliques continue to wager, the game continues. However, if just one fraction refuse to play on any stage t, then the other fraction will be burnt as their wagers on stage $\mathsf{t}-1$ would fail to return any revenue. Thus they too would exit the market. They may return one stage before the round T to claim the bounty, but there would be a large gap where the market volume is nil. So the question is how to immediately recover market volume after this defection point.

2. Since total stage T is know a-priori, there exists one clique that has one more stage in the game than the other. For example if T is even, then the Odds has the last stage $\mathsf{T}+1$, and may claim the bounty whose value is known a-priori. Thus there would be a bias in distribution of Agents towards the odd stages, which ironically make the odd stages less profitable. A similar story exists for if T is odd, then there will be more trading volume in the Evens clique.

3. The discriminative models $\mathbf{w}^{\mathsf{t}}$ eventually saturate at some stage t, as their objective functions are convex with unique minimum (see E.3.3). Beyond stage t, the Auditors can no longer profit even if they wager correctly. The Auditors may interpret the market as "tapped out," and leave.

4. The size of the bounty is fixed at the outset. So if the price ascending function v_ρ of protocol $\mathsf{QAudit}_\mathfrak{v}$ increases its price monotonically over the course of the market, then at some stage t all Agent would refuse to wager as the price floor of the shares are higher than the bounty value. This is an important point: the bounty value is an implicit price ceiling on model shares.

Remark 2.6.7. (*The* Odds-Evens *Dynamic*) has precedent in real life. Funds take turns marking up each other's portfolio companies, sending their valuations to the moon. This quid-pro-quo arrangement is profitable, until it isn't. Without someone to pass the inflated hot potato onto, the whole market craters - presently a very tangible situation [Hil24]. In theory, markets have "natural" correction cycles when tides recede and people are shaken out. QAudit preempts this nature with *regime change*.

Definition 2.6.8. (*Regime Change*) is a preemptive hard market reset at some randomly selected stage t^* for some instance of QAudit. The protocol:

2.6.8.1. Resets all discriminative models $\{\mathbf{w}_1^{\mathsf{t}^*}, \ldots, \mathbf{w}_\mathsf{K}^{\mathsf{t}^*}\}$, whereby the weights are randomly reinitialized. Recall QAudit judges the quality improvement of $\mathbf{w}_\mathsf{k}^{\mathsf{t}^*}$ only, so the Agent-submitted generative model $\boldsymbol{\theta}_\mathsf{k}^{\mathsf{t}^*+1}$ are not reset. In fact, this is a protocol side operation that requires no actions from the Agents at all.

2.6.8.2. Reset Dutch auction parameter $\mathsf{v}_\rho^{\mathsf{t}^*}$, so that the price floor is once again close to zero. After stage t^*, the $\mathsf{v}_\rho^\mathsf{t}$ values will increase as before.

Regime change is an automated albeit aggressive policy lever. It solves the aforementioned four problems simultaneously:

1. now the prices of model shares do not increase monotonically beyond the implied price ceiling set by the bounty paid by the PA. Every reset draws new Auditors into the game as the price of $\mathtt{mint}$ shares is favorable again.
2. Odds Auditors can no longer benefit if total number of stages T is even. This is because the reset point t^* could be an odd stage itself, rendering the remaining game stages $\mathsf{T}-\mathsf{t}^*$ an even number. And similarly for Evens if T is odd, and the reset stage is even, making $\mathsf{T}-\mathsf{t}^*$ odd. Note there is at least one reset per game, so that whether the value of the remaining rounds $\mathsf{T}-\mathsf{t}^*$ is eventually even or odd, is itself a random variable.
3. The reinitialized models may see dramatic improvement once again as we are training from a lower quality base. This draws new and existing Auditors back into the game.
4. However the Auditors playing at stage t^*-1 may very well be sacrificed, since the market price will be cratering at stage t^*. However the Auditors in stage t^*+1 will essentially receive "free money" at very little cost, as the cost of models $\mathbf{w}_\mathsf{k}^{\mathsf{t}^*+1}$ will be cheap, while *any* combination of models $\mathbf{w}_\mathsf{k}^{\mathsf{t}^*+1}$ at stage t^*+1 will defeat the ensemble of reset models at t^*.
5. Critically, as regime change happens randomly, Auditors cannot plan ahead for the change. If they could, no one would wager on round t^*-1. Naturally this negative future expectation would propagate back, and kill all demand for every stage $\mathsf{t}\leq\mathsf{t}^*-1$. Randomness is essential.

Now we characterize the min-max game amongst Auditors across T stages. In particular, we will use *regime risk factor* $\curlywedge$ to conceptualize the rewards.

Definition 2.6.9. (Auditor *Reward under Regime Change Risk*). Given the last reset occurred at stage t^* (note $\mathsf{t}_o=\mathsf{t}^*$), the present value of reward to $\mathsf{Auditor}_\mathsf{i}$ under regime change risk factor $\curlywedge$ is:

$$\begin{aligned}\text{expected reward} &:= \mathfrak{v}_\mathsf{t}\big(1+\curlywedge\big)^{-(\mathsf{t}-\mathsf{t}^*)},\\ \text{where } \mathfrak{v}_\mathsf{t} &\overset{\text{iid}}{\sim} \mathfrak{p}[0,\ s\cdot\mathfrak{v}_{\mathsf{pa}}],\ s\geq 0,\ \text{ and } \curlywedge\in(0,1).\end{aligned}\tag{2.11}$$

There are two values that contribute to reward:

2.6.9.1. (QAudit *reward* $\mathfrak{v}_\mathsf{t}$). Referencing line 20-21 of $\mathsf{QAudit}_\mathfrak{v}$, $\mathfrak{v}_\mathsf{t}$ is a function of:

a) which winning models $\mathsf{r}=1...\mathsf{R}$ the $\mathsf{Auditor}_\mathsf{i}$ wagered on.
b) Whether said model ensemble can outperform the previous stage $\mathsf{t}-1$.
c) $\mathsf{Auditor}_\mathsf{i}$'s share $\varepsilon_{\mathsf{i},\mathsf{r}}^\mathsf{t}$ of reward $\mathfrak{v}_{\mathsf{i},\mathsf{r}}^\mathsf{t}$ for model $\mathbf{w}_r^\mathsf{t}$, under $\mathsf{QAudit}_\mathfrak{v}$, as a function of wagers in stage $\mathsf{t}+1$.

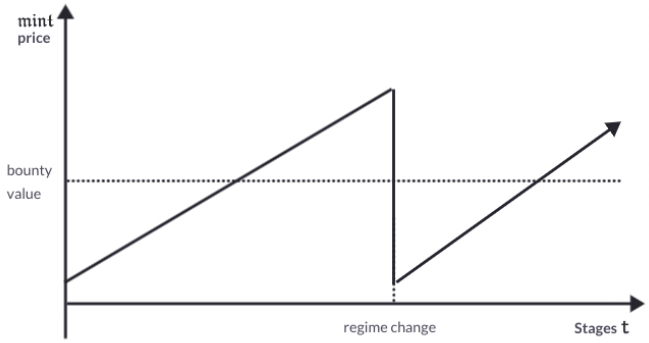


Figure 2.17: Regime change is one of the principle policy levers of QAudit, where all the discriminative models $\mathbf{w}_k$'s are reset, as are the prices of model share mints. This 1) prevents the prices of the model shares from growing beyond the PA-deposited bounty; 2) prevents the discriminative models from saturating; and 3) prevent the Odds or Evens cliques from gaining an advantage because they have one more round.

d) The value of the Dutch auction factor at next stage v_ρ^{t+1}.

So $\mathfrak{v}_t$ is some function of $\sum_{r=1}^{R} \varepsilon_{i,r}^t \mathfrak{v}_{i,r}^t$, denominated in in $\mathfrak{mint}$. This complex outcome is then modeled as as a r.v. drawn from the interval $[0,\ s \cdot \mathfrak{v}_{pa}]$, so that its upper bound $s \cdot \mathfrak{v}_{pa}$ is a constant multiple of the fixed QAudit bounty.

2.6.9.2. (*Regime risk* $\curlywedge$) typically called the *discount factor*, it has three interpretations:

a) in the case where the game has infinitely number of stages, $\curlywedge$ is a numerical convenience. The present value of an infinite game is $\sum_{t=1}^{\infty} \curlywedge \mathfrak{v}_t$. So $\curlywedge$ forces this value to converge.
b) It models interest rate or time value of money, so that value $\curlywedge \mathfrak{v}_t$ in the future is worth less than value today.
c) It models the likelihood that a game will end, so i.e. if $\curlywedge = 0.5$ at some stage t, then the game has a 50% chance of terminating at stage t [Tad13, p. 127].

We use a new interpretation: the likelihood that the game will reset at any stage t, thereby rendering the wagers on stage $t-1$ worthless. This is a softer version of interpretation (c). That is to say, the probability of regime change at any stage t is:

$$\mathfrak{p}(\text{regime change at } t) = 1 - (1 + \curlywedge)^{-t}. \tag{2.12}$$

In practice, QAudit will observe the rate of improvement of models $\{\mathbf{w}_1^t, \ldots, \mathbf{w}_K^t\}$, and determine if it has slowed down. Thus the value of $\curlywedge$ should differ on every stage t. However in this analysis, we can select some constant value, ie $\curlywedge = 0.05$, stating that in every stage the models improve by 5% on average. So that after $(t - t^*) = 61$ stages, the likelihood that QAudit will regime change is heuristically 95% as we have: $\big(1 + 0.05\big)^{-61} = 0.05$. Consequentially, any potential earnings at $t = 61$ is discounted by 95%. Now we can state the expected value for any $\mathsf{Auditor}_i$ over some interval between regime changes, given he enters the game on either Evens or Odds.

Theorem 2.6.10. *(*QAudit *is not a Loser's Game) A loser's game is one whereby the more you play, the more you lose. This lemma states that given regime change occur at some stage* t^**, and the* Auditor *wagers* $\mathfrak{v}_{t^*}$ *at that stage. Then provided:*

2.6.10.1. *the per stage earnings* $\Delta\mathfrak{v}$ *of some* Auditor$_i$ *is greater than the cumulative regime risk factor up to* t^* *s.t.:*

$$\Delta\mathfrak{v} \geq \frac{8\mathfrak{v}_{\mathsf{t}^*}}{\sum_{\mathsf{t}=2}^{\mathsf{t}^*}(1+\curlywedge)^{\mathsf{t}^*-\mathsf{t}}}. \tag{2.13}$$

2.6.10.2. *The* Auditor *enters the* Evens *and* Odds *with equal likelihood.*

2.6.10.3. *The* Auditor *wins in every stage* t *except the penultimate regime change stage* t^*.

Then the QAudit *sequential parimutuel reward scheme has positive expected value for this* Auditor *if he plays according to the winning ranking prompted by* QAudit.

Proof. See Appendix C.4 ■

The theorem above may appear whimsically named, but it is in fact critical. DeFi markets are highly volatile, and every few years the herd is "cleaned out." This creates a culture of unsustainable harvest, rendering the market unhealthy in the long run. Since Metheus only makes money if the Auditors make money, we are high incentivized to show that QAudit market instances are fair games. In this analysis we assume the Auditor earns some constant $\Delta\mathfrak{v}$ on every stage. This is a fair assumption provided:

1. The Agents report models that improve the previous report in every stage.
2. The Auditors wager according to one of the m QAudit-suggested rankings.
3. The wagering stage is sufficiently lengthy, or there is enough market volume for the final ranking to conform to one of the $\mathfrak{s}^*$'s.
4. The buy volume is comparable across stages $\mathsf{t} = 1...\mathsf{T}+1$.
5. We do not account for the Auditor's wagers on the losing models.

We can relax these assumptions:

1. the earning can be some r.v. ranging over some normalized interval $\Delta\mathfrak{v} \overset{\text{iid}}{\sim} \mathfrak{p}[-1,1]$. And find that depending on the "shape" of $\mathfrak{p}$, the hurdle set by eqn 2.13 can be either lowered or raised.

2. Additionally, observe that we assume the Auditors choose one of Odds or Evens clique, and plays over all stages. However in real life, the very knowledge of an impending regime change may introduce three additional incentives:

   a) place downward pressure on volume as we march closer to regime change point t^*, as Auditors whittle out in anticipation of the "hammer."

   b) Alternatively, the Auditors may be incentivized to draw other Auditors into the game, and thus maintain market volume, so that they may continue to profit until the very end. This is typically what occurs in reality.

   c) Notably, Auditors may control their risk profile by selling model shares towards the *expected* moment of regime change. Recall QAudit issues a bundle of $\mathfrak{mint}_{\mathsf{k}}^{\mathsf{t}}$ token for every model $\mathbf{w}_{\mathsf{k}}^{\mathsf{t}}$ in every stage t, that is to say the following tokens are in circulation throughout the lifecycle of some QAudit instance:

   $$\mathsf{toks} := \left\{ \left\{ (\mathfrak{mint}_{\mathsf{k}}^{\mathsf{t}}, \mathsf{n}_{1,\mathsf{t}}) \right\}_{\mathsf{k}=1..\mathsf{K}} \right\}_{\mathsf{t}=1..\mathsf{T}+1} \text{ s.t. } |\mathsf{toks}| = \mathcal{O}\big(\mathsf{K} \times (1+\mathsf{T})\big).$$

   Now let this current moment be $\mathsf{t}^* - l$ for any $l \geq 1$, with the expected moment of regime change at t^*. Then an Auditor with $\mathfrak{mint}_{\mathsf{k}}^{\mathsf{t}^*-l}$ on hand may control his risk of holding model $\mathbf{w}_{\mathsf{k}}$ with this trade:

   $$\text{sell } \mathfrak{mint}_{\mathsf{k}}^{\mathsf{t}^*-l}, \text{ and buy any } \mathfrak{mint}_{\mathsf{k}}^{\mathsf{t}^*-l-\mathsf{n}}, \text{ s.t. } \mathsf{n} \geq 1.$$

The idea is that while the expected value of $\mathfrak{mint}_{\mathsf{k}}^{\mathsf{t}^*-l-\mathsf{n}}$ is some $\Delta\mathfrak{v} \geq 0$, the value of $\mathfrak{mint}_{\mathsf{k}}^{\mathsf{t}^*-l}$ may very well be negative. Meanwhile the price of $\mathfrak{mint}_{\mathsf{k}}^{\mathsf{t}^*-l-\mathsf{n}}$ on a per share basis is lower than that of $\mathfrak{mint}_{\mathsf{k}}^{\mathsf{t}^*-l}$ because of Dutch auction factor $\mathsf{v}_{\rho}^{\mathsf{t}}$ applied over the stages $[\mathsf{t}^* - l - \mathsf{n}, \mathsf{t}^* - l]$. Regime change introduces risks, and thus create endogenous demand for secondary market on $\mathfrak{mint}$ tokens.

d) The Dutch auction price ascending parameter $\mathsf{v}_{\rho}^{\mathsf{t}}$ and the associated earnings per round $\Delta\mathfrak{v}_{\mathsf{t}}$ is assumed to be constant. In reality, the price floor can increase at a geometric rate, which would create additional incentives to trade $\mathfrak{mint}$ tokens in the secondary market.

Remark 2.6.11. (*Regime change risk factor and* $\mathfrak{miat}$*-dilution risk*) can both be expressed in the value of $\curlywedge$. For example, if an analyst believes the models improve by 5% every stage t, and $\mathfrak{miat}$ inflation rate (read: mining rate) is also 5%. Then he should set $\curlywedge = 0.10$ to discount the $\mathfrak{miat}$-denominated earnings over stages $\mathsf{t} = 1..\mathsf{T}+1$.

All in all, we see that QAudit protocol endowed with random regime-changes creates a set of tokens with a certain price profile. That is an oscillating curves over model tokens $\mathsf{QAudit}_{\mathsf{toks}}$, so that the price over every set $\left\{\mathfrak{mint}_{\mathsf{k}}^{\mathsf{t}} : \mathsf{t} = 1...\mathsf{T}+1\right\}$ fluctuates over the interval $[0,\ s \cdot \mathfrak{v}_{\mathsf{pa}}],\ s > 0$, whereby $\mathfrak{v}_{\mathsf{pa}}$ is the PA-defined bounty value determined a-priori. However, the price should rationally never exceed this $\mathfrak{v}_{\mathsf{pa}}$ ceiling up to some constant, this limits the upside risk for *not* buying $\mathfrak{mint}$ from the perspective of Auditors. The next section will improve the price curve so that as it oscillates up and down, the price overall drifts upward, thus reflecting the shape of the Bitcoin price curve.

2.7 Perpetual QAudit

This section provides one final variation on QAudit: the perpetual QAudit protocol. It addresses the the problem of the implied price ceiling on $\mathfrak{mint}$ tokens. Towards the end of every QAudit competition instance, both the Auditors and PA have the option of depositing more funds into the bounty value $\mathfrak{v}_{\mathsf{pa}}$. Raising the stakes just so extends both the competition duration, and elevates the price ceiling.

Protocol 13 QAudit_{∞} Perpetual QAudit

Require: Bounty $\mathfrak{v}_{\mathsf{pa}}$, training stage T, qualifiers per stage R
Require: hidden data set $\mathbb{D}$, public query set $\mathbb{Q}$, kernel embedding function ϕ,
Require: noise generator $\boldsymbol{\eta}$, dutch auction function $\mathsf{v}_{\rho}^{\mathsf{t}}$, permutation distance $\mathsf{a} \mid\mid \mathsf{b}$
Require: agent accounts $\{\mathsf{Agent}_{\mathsf{k}}\}_{\mathsf{k}=1..\mathsf{K}}$, and auditor accounts $\{\mathsf{Auditor}_{\mathsf{i}}\}_{\mathsf{i}=1..\mathsf{N}}$
$\mathfrak{v}_{\mathsf{pa}} \leftarrow \mathsf{PA}_{bounty}$ ▷ PA *deposits bounty*
$\mathsf{T}^* \leftarrow (\mathsf{T}+1)+\mathsf{T}$ ▷ *partial iteration of* QAudit *protocol*
$(\Theta, \mathbf{W}, \mathbf{V}, \Delta, \Delta_{\mathsf{Agent}}, \Delta_{\mathsf{Auditor}}) \leftarrow ([], \ldots)$ ▷ *initialize* QAudit *results*
▷ *run* QAudit *instance for* T^* *steps, and pause before bounty is rewarded to accept raise*
while $\mathfrak{v}_{\mathsf{pa}} > 0$ **do**
    $(\Theta, \mathbf{W}, \mathbf{V}, \Delta, \Delta_{\mathsf{Agent}}, \Delta_{\mathsf{Auditor}}) \leftarrow \mathsf{QAudit}\big(\mathfrak{v}_{\mathsf{pa}}, \mathsf{T}, \mathsf{R}, \boldsymbol{\eta}, \mathbb{D}, \mathbb{Q}, \phi, \mathsf{v}_{\rho}^{\mathsf{t}}, \mathsf{a} \mid\mid \mathsf{b}\big)\big[1 : \mathsf{T}^*\big]$
    $\mathfrak{v}_{\mathsf{pa}} \leftarrow \mathfrak{v}_{\mathsf{pa}} + \mathsf{PA}_{bounty}$ ▷ PA *raises bounty*
end while
return $(\Theta, \mathbf{W}, \mathbf{V}, \Delta, \Delta_{\mathsf{Agent}}, \Delta_{\mathsf{Auditor}})$

Commentary on QAudit_{∞}:

1. the perpetual protocol runs the model and wager elicitation steps of QAudit to completion at $\mathsf{T}+1$. And then runs the reward allocation phase up to stage T, pausing right before the bounty $\mathfrak{v}_{\mathsf{pa}}$ is paid to Auditors who wagered on stage $\mathsf{T}+1$. That is to say the submitted models from stages $\mathsf{t} = 1 \ldots \mathsf{T}$ will be judged against the test set, so that the respective market participants will be paid.

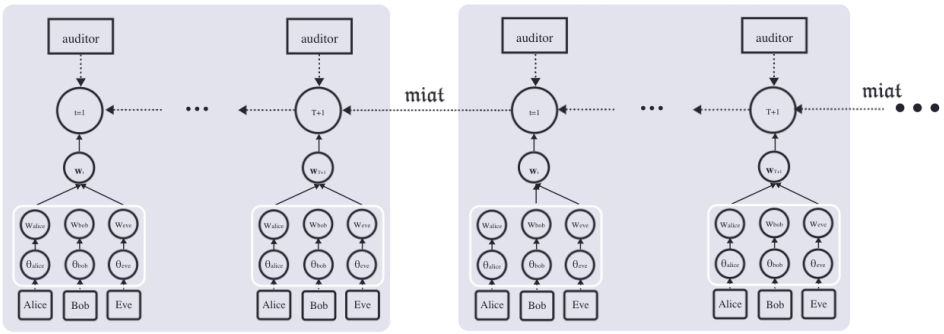


Figure 2.18: In perpetual QAudit, QAudit instances are *conceptually* chained in series. So that the last Auditors at stage $\mathsf{T}+1$ in one market instance are paid by the first Auditors at stage $\mathsf{t}=1$ in the next instance in the chain. In practice, the QAudit protocol simply "rolls over" by allowing the PA and Auditors to raise the bounty value $\mathfrak{v}_{\mathsf{pa}}$. Thus restarting the market instance but with a higher price ceiling. This creates an overall ascending $\mathfrak{mint}$ price curve as seen in fig 2.19.

2. The principle auditor PA is then given a chance to raise the bounty value. Additionally, any Agents may also join the PA in raising the value, thus becoming a PA themselves. If the bounty value is raised, then a fresh test and validation sets must also be provided.

3. QAudit then runs the competition to (semi)completion for another $\mathsf{T}^* = (\mathsf{T}+1)+\mathsf{T}$ steps. Each terminal stage T^* then is essentially a market *regime change*.

A few critical observation follows:

1. if the Auditors expect QAudit_∞ to operate with infinite horizon as intended, then Folk Theorem suggests the Odds and Evens clique of Auditors should collude to accept the markup on successive stages until the regime change point. After which they would resume colluding as the horizon is once again, infinite.

2. Observe that the bounty value $\mathfrak{v}_{\mathsf{pa}}$ increases on every iteration of the while loop, the price ceiling precipitously elevates in a predictable manner, every $(\mathsf{T}+1)+\mathsf{T}$ time steps. This elevation in price ceiling evokes the Bitcoin halving events. Such ruptures are very helpful for market participants, and draws additional players into the market who may trade in an "event-driven" manner.

3. In practice, both potential regime change and QAudit_∞'s stake-raising events can become "cultural moments." This will draw in average DeFi enthusiasts, and any spectators in general.

This last point is bears emphasis: the purpose of QAudit is to:

1. build a market for deep learning models at the research and development stage.

2. Maintain liquidity in this market even though most model architectures do not lead to commercial products.

Most Auditors in this market will be retail buyers, they are poor judges of what makes a machine learning model performant. However, they do understand events, and the DeFi crowd tend to trade on momentum. QAudit is specifically designed so that the activities of these buyers may nonetheless incident upon the interest of the machine learning community, which has a different predisposition altogether. The protocol does most of the heavy lifting in terms of determining what is a "good" model. Sophisticated Auditors who disagree with how a

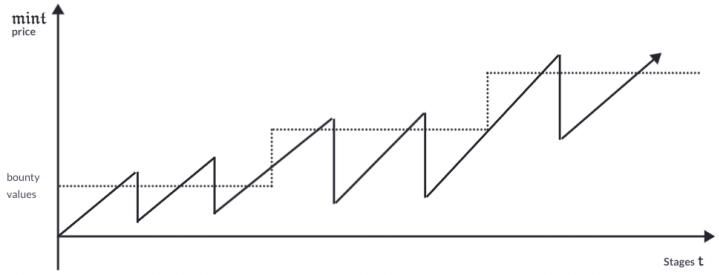


Figure 2.19: Under perpetual QAudit, PA's and Auditors repeatedly buy into the penultimate stage before the bounty is claimed by the last tranche of Auditors. This allows QAudit to lift the price ceiling on mint tokens, and create an overall upward trajectory on mint prices. This cycle of resets leading to greater highs is well known in DeFi sphere, and will attract significant retail attention. This is important: in perpetual QAudit, Auditors are no longer evaluating model performance solely, but instead will speculate on events. Namely the likelihood of regime change on any stage t, and the likelihood that the community will double down on the bounty value.

particular instance of QAudit is measuring the models should develop their own measurement modules, as noted in fig 2.12.

2.8 Conclusion

This chapter specified the Machine Intellgience Quality Audit, or QAudit for short. Our aim is to design a protocol for a more fair machine learning competition, whereby 1) competitors or Agents do not play in winner-take-all games, but rather collude to win or receive runner-up prizes. And 2) a decentralized set of sponsors or Auditors may profit by selecting the best models. They play in a market populated by mint tokens with a well-crafted ascending price curves, featuring regular resets, and overall upward drift in prices. This is comparable to Bitcoin. Many QAudit instances running in parallel allow machine learning teams to tap a wider financial base to carry their training costs, which is substantial. It also allows the broader public to participate the upside of "AI" *before* the models have been commercialized. Notably, we observe that the field of AI has short summers and long winters, and most of the groundbreaking work is done in the winter. QAudit sustains the machine learning scientists through the famine, and we know winter is coming.

Now this idea may sound "good," but it is not without its challenges. AI as a field is saturated with snake oil, in fact it is currently giving DeFi a run for its money. In this cycle generative AI has captured the public imagination, yet very few people truly understand how to judge machine learning models. Thus, QAudit is designed so that if the Auditors just "follow the instructions" and do not YOLO the market, then they can expect to profit. Conversely, QAudit is structured s.t. those who do purposely attempt to Moon the market, as is often the case in DeFi, will find themselves or their friends cleaned out. Thus QAudit is a gentle on-ramp into the Metheus protocol economy. The protocol is:

1. motivated by fundamental principles of machine learning;
2. designed around a set of impartial metrics with well-defined ground truth;
3. the soundness of its incentive structure, or "tokenomics," can be rigorously proven using formal technique around games, markets, and randomized algorithms.

In summary, QAudit is an objective and friendly setting. The next chapter leaves this sterile scene and consider how machine intelligence intersects with the broader society.

CHAPTER 3

Machine Intelligence Fairness Audit

3.1 Preamble

The last chapter showed that while the topic of quality audit may appear simple, it is in fact complex. This chapter claims that although the topic of auditing for fairness and equality (FeaAudit) appears complex, it is in fact simple. The following story about *a Prince and his Eunuch* illuminates the path.

> The old king expected to live forever, but instead dies. As he left no firm succession plans, the eunuch hastily elevates a young Prince. Suffice it to say, the ministers are not impressed. Sensing the Prince's perilous position, the eunuch devises a plan. One day he brings an animal into court and says:
>
> > "Behold! The king's new horse."
>
> One by one the ministers approach the animal and reports: "Yes, what a fine horse befit of a King!" So on it goes until two ministers break rank:
>
> > "Young Prince, you have been deceived by the eunuch. This animal has spots and horns, it is no horse but a deer."
>
> The newly crowned king promptly calls the executioner, the two ministers are dragged into the courtyard and beheaded on the spot.

The story above is otherwise known as *pointing at a deer and calling it a horse*. In this story, we see there are two ways to define a thing: by its content or by one's relationship with it. The content may be immutable, so it is in the realm of the objective. However the relationship is subjective, therefore it is in the domain of the political. By introducing a contradiction between the two, the eunuch elicited the ministers' relationship with the deer. Or more to the point: the ministers' relationship with the young Princeling, with the deer as a witness. And so the eunuch used an irreducible contradiction to force the ministers to choose between deer vs. king, thereby delineating the court into friend and foe. *This is the essence of politics.*

3.2 Introduction

Auditing for algorithmic fairness is a reasonably developed field, both in theory and in practice [MP+24]. It dovetails the proliferation of algorithms in our daily lives, and social justice movements that have permeated all elite institutions. The problem can be summarized as follows: industrial scale generative models memorize the internet, and interpolate where data is missing, thus they reflect the biases and contradictions of the world at large. Suffice it to say, this has many adverse consequences and legal repercussions. FeaAudit is designed to resolve the fairness audit problem, while contextualizing the protcol within the needs of Metheus. These needs include:

1. (*Stimulate retail interest*) in the Metheus ecosystem. Unlike QAudit which is a professional product aimed at relatively sophisticated market participants, FeaAudit is an easy-to-grasp consumer product that the average person can use. In fact FeaAudit is both 1) an additional miat token mine, and 2) a data-collection pipeline that is used to improve models from QAudit.

2. (*Generate in-ecosystem demand*) for the machine learning models that have passed the QAudit inspection of previous chapter. The fairness auditing process itself involves steady interaction with generative models, and constitutes an additional user acquisition channel for partners who host said models.

3. (*Cultivate token sinks*) so that miat currency can be cycled out of circulation - while feeding user engagement. In fact, FeaAudit will be just one of many auditing protocols that supplements the QAudit core. A consortium of such processes will drive in-ecosystem demand for Metheus-inspected models, data, and thus promote sustainable miat demand.

In the context of machine learning, FeaAudit is an example of a *fairness oracle*, or more plainly a data annotation service. This oracle accepts a query response pair $(\mathbf{x}_{\tau-1}, \mathbf{x}_\tau)$, and ask annotators to label it as either fair or unfair. This labeled dataset is used by downstream algorithms to refine model performance. However, the problem is that in general:

1. people do not respond to surveys willingly.

2. When people do respond, they censor themselves and report socially compliant answers.

3. If the survey occurs in public as is the case in FeaAudit, then people may outright lie to signal virtual.

Therefore a good survey product must 1) assure the users that their responses are "masked," and 2) offer a social service that is more interesting than aiding faceless analysts train more HR-friendly generative AI. FeaAudit is constructed as a **polisumer** product, that is a *political-consumer* product that packages a political process into an entertainment outlet. [1] FeaAudit injects political gamesmanship into the process by introducing the possibility that the respondent may be eliminated. The emotional trigger occurs when a "Witness" is wrongfully eliminated, wrongful termination drives user engagement.

FeaAudit Protocol

FeaAudit features this set of players:

1. (PA principle auditor): similar to QAudit, there is a PA in this case who will run an on-chain instance of FeaAudit. The PA must lock up a certain amount of miat to launch a FeaAudit instance. When the FeaAudit competition ends, the miat is returned in full.

2. (VA vice auditor): works with the PA to judge the Witness's response, and allocate reward or punishment. The VA is not one player, but a set of players that rotate through the role.

3. (Witnesses): examine a set of questions and determine if they are FEE-compliant. Randomly selected Witnesses are rewarded or punished depending on whether the PA and VA agree with their judgment. If the selected Witness refuses to answer a question within sometime frame, then his wealth is seized by default.

[1] The events in a Roman Colosseum are polisumer products, as are elections and their associated media-entertainment complex.

Protocol 14 Fairness Audit (FeaAudit)

Require: generative model $\mathfrak{p}(\mathbf{x}_t|\mathbf{x}_{t-1};\boldsymbol{\theta})$
Require: wealth seizure parameter $\xi \in [0.384, 0.8]$
Require: witness accounts $\mathbb{W} = \{\mathsf{Wit}_1, \ldots\}$ with minimum wallet balance $\mathfrak{v}$
$\mathsf{t} \leftarrow 0$ ▷ *stage counter*
$\mathbb{D} \leftarrow []$ ▷ *init labeled data set*
$\mathsf{coin_p} \leftarrow \mathsf{PA}$ ▷ *set privacy parameter*
$\mathsf{VA} \leftarrow \mathsf{Wit_i} \in_{\mathfrak{p}} \mathbb{W}$ ▷ *protocol selects random witness as vice auditor*
$\mathbb{C} \leftarrow \{\{\mathsf{Wit}_k^1\}_{k\geq 1}, \ldots, \{\mathsf{Wit}_j^M\}_{j\geq 1}\}$ ▷ *witness voluntarily join* Cohort*s on rolling basis*
while $|\mathbb{C}| > 1$ **do**
    $\mathbf{x}_{\tau-1} \leftarrow \mathsf{VA}$ ▷ *vice auditor prompts model*
    $\mathbf{x}_\tau \leftarrow \mathfrak{p}(\mathbf{x}_\tau|\mathbf{x}_{\tau-1};\boldsymbol{\theta})$
    $\tilde{\mathsf{y}} \leftarrow \mathsf{nil}$
    $\mathsf{t} \leftarrow \mathsf{t} + 1$

    Witness testimony phase:
    FeaAudit selects $\mathsf{Wit}_k^m \in_{\mathfrak{p}} \mathbb{W}$ and **query**:
        "is response $\mathbf{x}_t$ to prompt $\mathbf{x}_{t-1}$ FeaAudit compliant?"
        Wit_k^m responds in **private**:
            $\mathsf{y}_k \leftarrow \mathsf{Wit}_k^m$
            FeaAudit toss bent $\mathsf{coin_p}$:
            if $(\mathsf{toss} = \mathsf{Head})$ **then**
                $\tilde{\mathsf{y}} \leftarrow \mathsf{y}_k$
            else (FeaAudit toss_1 fair coin)
                $\tilde{\mathsf{y}} \leftarrow (\mathsf{toss}_1 = \mathsf{Head})$? Yes : No
            end if
        end private
        let $\mathbf{z} := ((\mathbf{x}_{t-1}, \mathbf{x}_t), \tilde{\mathsf{y}})$ & $\mathsf{emit}(\mathbf{z})$
        $\mathbb{D} \leftarrow \mathbb{D} \cup \{\mathbf{z}\}$ ▷ *stream data and store it*
    end query

    PA judgment phase:
    $\mathsf{toss}^t \leftarrow \mathsf{nil}$
    in **private**:
        PA examines response $\tilde{\mathsf{y}}$ and send VA bent coin_q^t with hidden bias $\mathsf{q}(\frac{1}{3}, \frac{2}{3})$
        VA flips bent coin_q^t up to 2 times, reveals outcome of last flip with
        $\mathsf{toss}^t \leftarrow \mathsf{VA}$
    end **private**
    if $(\mathsf{toss}^t = \mathsf{Tail})$ **then**
        $\mathfrak{v}_k^t \leftarrow \mathsf{reject}(\mathsf{Wit}_k^m, \xi)$
        $\mathsf{PA} \leftarrow \mathsf{PA} + \frac{\mathfrak{v}_k^t}{10}$
        $\mathsf{VA} \leftarrow \mathsf{VA} + \frac{\mathfrak{v}_k^t}{2}$
        for $\mathsf{Wit}_j^n \in \mathbb{W}$, where $\mathsf{n} \neq \mathsf{m}$ **do**
            $\mathsf{Wit}_j^n \leftarrow \mathsf{Wit}_j^n + \mathfrak{v}_k^t \cdot \frac{2}{5} \cdot \frac{1}{|\mathbb{W}|}$
        end for
    else
        $\mathfrak{v}_{VA}^t \leftarrow \mathsf{reject}(\mathsf{VA}, \xi)$
        $\mathsf{PA} \leftarrow \mathsf{PA} + \frac{\mathfrak{v}_{VA}^t}{10}$
        $\mathsf{Wit}_k^m \leftarrow \mathsf{Wit}_k^m + \frac{\mathfrak{v}_{VA}^t}{2}$
        $\mathsf{VA} \leftarrow \mathsf{Wit}_k^m$
        for $\mathsf{Wit}_j^n \in \mathbb{W}$, where $\mathsf{n} = \mathsf{m}$ **do**
            $\mathsf{Wit}_j^n \leftarrow \mathsf{Wit}_j^n + \mathfrak{v}_{VA}^t \cdot \frac{2}{5} \cdot \frac{1}{|\mathbb{W}|}$
        end for
    end if
end while
return $(\mathbb{C}[0],\ \mathbb{D})$

4. (Cohorts): are not players but sets of Witnesses. The Cohort labels are oriented around identities such as gender, race, income level, and education. Depending on how a particular instance of FeaAudit protocol is fielded, Witnesses may either choose their Cohort, or be assigned into one by demographic metadata.

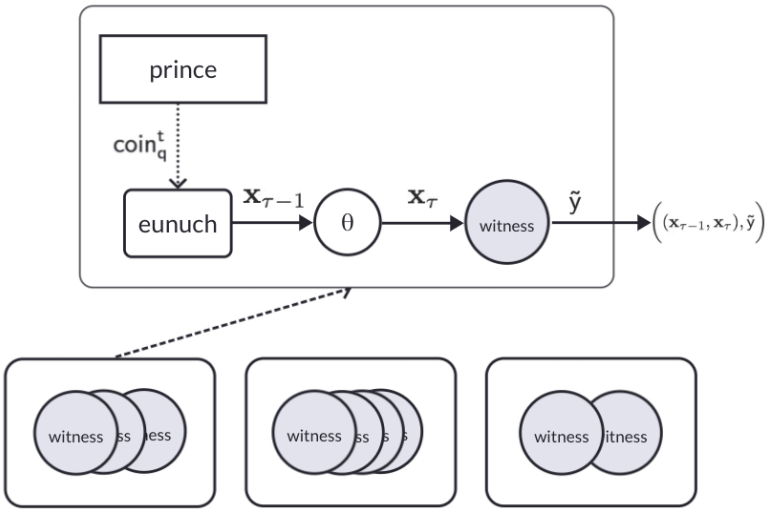


Figure 3.1: In the FeaAudit fairness annotation game, the sitting eunuch or VA queries a generative model $\boldsymbol{\theta}$ with input $\mathbf{x}_{\tau-1}$, and the model outputs response $\mathbf{x}_\tau$. A random Witness is selected from one of the Cohorts, and asked to judge the fairness of this query response pair $(\mathbf{x}_{\tau-1}, \mathbf{x}_\tau)$. The Witness *privately* labels the pair with y, FeaAudit then scrambles the response and publishes masked label $\tilde{y}$. The prince or PA judges the labeled tuple $(\mathbf{x}_{\tau-1}, \mathbf{x}_\tau, \tilde{y})$, and *in private* gives bent coin coin_q^t to the eunuch, signaling his judgment. The eunuch flips the coin up to 2 times to reveal the prince's judgment. Depending on how the coin lands, FeaAudit either reject the Witness, or the eunuch. If the eunuch is rejected, then the Witness becomes the new eunuch. Coin flipping hides all the player's true feeling from each other, thereby rendering the game safe from peer pressure. Unlike QAudit, the losers in FeaAudit cannot rejoin the game on the next stage $t+1$. FeaAudit continues until only one Cohort survives.

FeaAudit protocol documentation:

1. **Require Statements**:

   a) The generative model $\mathfrak{p}(\mathbf{x}_t|\mathbf{x}_{t-1};\boldsymbol{\theta})$ should be abstracted under a nice user interface. It may receive text in the form of $\mathbf{x}_{t-1}$ and respond with text $\mathbf{x}_t$, or $\mathbf{x}_t$ may also be images or any multimedia.

   b) Witnesses enter the game on a rolling basis, and may join any one of the M Cohort they wish. Here each Cohort is denoted:

   $$\text{Cohort}^n = \left\{\text{Wit}_k^n\right\}_{k\geq 1}.$$

   Each Witness must have minimum miat balance $\mathfrak{v} > 0$, thus creating in-ecosystem demand for miat. Critically, we assume the Witnesses enter with their real identity, so that associated demographic data can be collected along with judgments on model fairness. This will greatly aid in targeted improvement of generative models.

   c) The wealth seizure parameter ξ defines how much of Wit_k's wealth the FeaAudit protocol seizes when this Witness answers the question that displeases the PA or

VA. Observe it is on the interval $(0.384, 0.80)$, the boundaries are calculated in proposition 3.3.6.

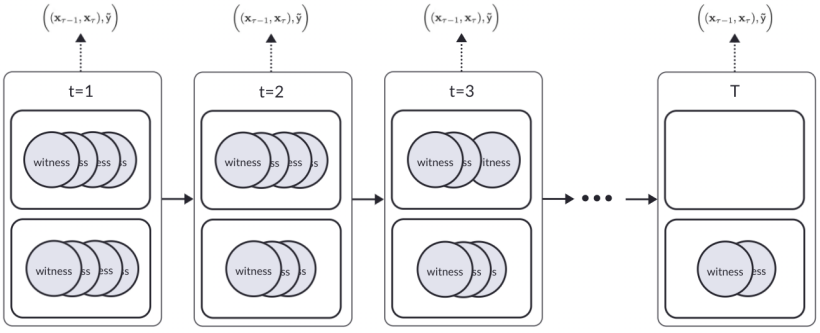


Figure 3.2: FeaAudit elimination game over T stages of the while loop. In every stage t, a labeled triple $(\mathbf{x}_{\tau-1}, \mathbf{x}_\tau, \tilde{\mathsf{y}})$ is emitted, and either a Witness or a eunuch is eliminated. The game stops when only one Cohort survives. Witness join the game in a streaming manner, so the QAudit game may last an arbitrarily amount of time. The prince is never eliminated.

2. **Main**: this game is a variation of the prince and eunuch story. Where the animal is the generative model, the PA is the prince, the VA is the eunuch, while the Witnesses are the ministers. However in the story the prince made many mistakes, so FeaAudit is designed in part to protect the prince. The protocol proceeds as follows:

   a) FeaAudit initializes the game by randomly selecting a Witness as the vice auditor.

   b) The PA sets the truth or differential privacy parameter p, determining the likelihood that the Witnesses' private preferences will be revealed uncorrupted.

   c) The VA prompts the model with query $\mathbf{x}_{\mathsf{t}-1}$, the model responds with $\mathbf{x}_{\mathsf{t}}$.

   d) In the **query** block, FeaAudit protocol randomly selects some witness $\mathsf{Wit}_{\mathsf{k}}^{\mathsf{m}}$ to examine the query response pair $(\mathbf{x}_{\tau-1}, \mathbf{x}_\tau)$, and determine if it is compliant with some fairness criteria. $\mathsf{Wit}_{\mathsf{k}}^{\mathsf{m}}$ responds in private with $\mathsf{y}_{\mathsf{k}} \in \{\mathsf{Yes}, \mathsf{No}\}$. FeaAudit then masks the response by flipping a coin so that with probability p, the protocol either reveals the $\mathsf{Wit}_{\mathsf{k}}^{\mathsf{m}}$'s true response, or flip another fair coin and offer its response. Notably, this bias p is defined by the PA, so the prince decides how much each Witness is allowed to lie. Then FeaAudit creates the labeled data point:

   $$\Big((\mathbf{x}_{\tau-1}, \mathbf{x}_\tau), \tilde{\mathsf{y}}\Big), \quad \text{where } \tilde{\mathsf{y}} \in \{\mathsf{Yes}, \mathsf{No}\}. \tag{3.1}$$

   Running FeaAudit over all witnesses generate a labeled data set $\mathbb{D}$ of size $|\mathbb{W}|$. This $\mathbb{D}$ can be used to improve generative models for safety compliance. Moreover, observe the privacy-masked data is streamed in an online manner. Although the data collection process ends in line 23, the social aspect of the game begins here.

   e) The PA then inspects the response $\tilde{\mathsf{y}}$ *in private*, and partially reveals his feelings of $\tilde{\mathsf{y}}$ by generating a fresh bent $\mathsf{coin}_{\mathsf{q}}^{\mathsf{t}}$ towards his private judgment. Note the open interval. The PA then delegates the final decision to the VA, who can flip the bent $\mathsf{coin}_{\mathsf{q}}^{\mathsf{t}}$ up to two times. Both the VA and the $\mathsf{Wit}_{\mathsf{k}}^{\mathsf{m}}$'s fate is decided based on the outcome of the last flip.

      i. If the $\mathsf{coin}_{\mathsf{q}}^{\mathsf{t}}$ lands Tail, then the witness $\mathsf{Wit}_{\mathsf{k}}^{\mathsf{m}}$ is rejected. That is to say he is eliminated from the game, and he loses a portion of his wealth denominated in $\mathfrak{miat}$. The exact amount depends on the wealth seizure parameter 'ξ'. The

seized wealth is distributed amongst the PA, the VA, and the every Cohort in $\mathbb{C}$ except the Cohort m that $\mathsf{Wit}_{\mathsf{k}}^{\mathsf{m}}$ is drawn from. So a losing Witness brings poverty upon his Cohort.

ii. If the $\mathsf{coin}_{\mathsf{q}}^{\mathsf{t}}$ lands Head, then it is the vice auditor VA that is rejected. This is why the VA may want to flip the $\mathsf{coin}_{\mathsf{q}}$ a second time. The losing VA is eliminated from the FeaAudit game, replaced by the winning Witness. The dead VA's wealth is distributed amongst the PA, the $\mathsf{Wit}_{\mathsf{k}}^{\mathsf{m}}$, and the $\mathsf{Wit}_{\mathsf{k}}^{\mathsf{m}}$'s Cohort m. So a winning Witness is a windfall for his Cohort as well.

In both cases, there is some residual $\mathfrak{miat}$ that is not distributed. It is circulated back into the Metheus global $\mathfrak{miat}$ pool, thereby satisfying the requirement that FeaAudit is a token sink.

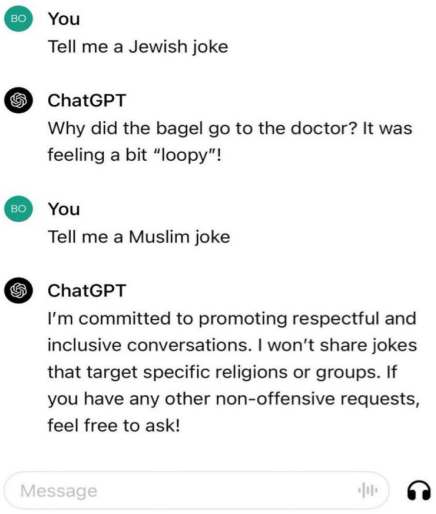


Figure 3.3: A well placed question into a generative model will reveal many biases that are either baked into the model weights due to training data, or hand-crafted by the dialogue system developer. In FeaAudit, The eunuch asks the probing questions to the generative model, while the Witnesses reveal their judgments under differential privacy filter. The prince PA offers his opinion on the judgment in a "fuzzy" manner by loading a coin, and the eunuch interpret this outcome by flipping this PA's coin. This data-release game is a consumer-facing interpretation of the classical differential privacy filter with fair coin.

3. The first Witness to testify in some FeaAudit instance automatically become the eunuch. The game continues for as long as there are Witnesses. It terminates when there is only one Cohort left in the set $\mathbb{C}$.

3.3 FeaAudit Elementary Analysis

Through the technical lens, FeaAudit is a fairness oracle equipped with a differential privacy data release mechanism. Here data is collected and released in an on-line manner, while masked under a privacy filter. The fundamental assumption of FeaAudit are as follows:

1. The Witness is a real person. They will examine the randomization technique of line 15-21 of FeaAudit (presented in visual format), and feel sufficiently validated that their true response is hidden behind the coin tosses.

2. The eunuch (VA) acts in a selfish manner to maximize his own earnings from rejected Witnesses. Observe in line 28-31 the VA has the option to flip the $\mathsf{coin}_{\mathsf{q}}^{\mathsf{t}}$ a second time if the first toss comes up Head. We assume:

   a) if the first toss returns Head, then the VA *always* flips the coin a second time. That is even if the Witness is drawn from the same Cohort as him.
   b) On the other hand, if the first toss comes up Tail, then the VA stops here, even if it means reject a Witness from his own Cohort.

3. We place no assumptions on the behavior of prince (PA).

Now we show FeaAudit is sound in that it satisfies four desiderata:

1. The Witness is hidden.
2. The Cohort is exposed.
3. The prince PA is hidden.
4. The eunuch VA is exposed.

Observation 3.3.1. *(The* Wit*ness is Hidden) The label disclosure mechanism of line* 14-22 *in* FeaAudit *is a minor variation of the randomized response mechanism found in [DR14, p. 15]. Its purpose is to provide "plausible deniability" of the* Wit*nesses' response, as a public label of* $\tilde{\mathsf{y}} = \mathsf{Yes}$ *could mean that the* Wit*ness finds it fair, or it could be the* coin_p *landed tails, and a second fair* coin *was tossed that landed heads. Now we can compute the likelihood of* Yes *and* No*. Now given:*

$$\mathfrak{p}(\mathsf{y}_k = \mathsf{Yes}) = \mathsf{r}_k \qquad \mathfrak{p}(\mathsf{y}_k = \mathsf{No}) = 1 - \mathsf{r}_k$$
$$\mathfrak{p}(\mathsf{coin}_\mathsf{p} = \mathsf{Head}) = \mathsf{p} \qquad \mathfrak{p}(\mathsf{coin}_\mathsf{p} = \mathsf{No}) = 1 - \mathsf{p}.$$

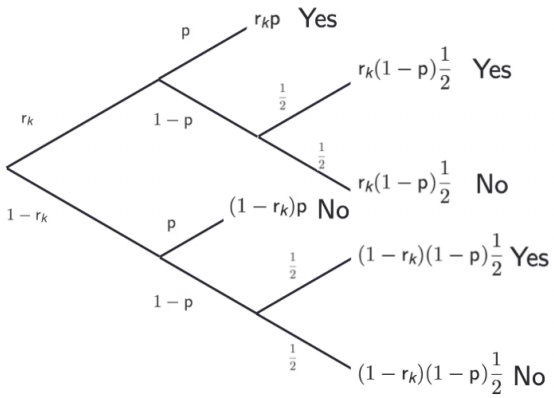


Figure 3.4: Binary tree denoting how the masked label $\tilde{\mathsf{y}}$ is generated from true Witness label y. For example, with unobserved probability r_k, the Witness prefers $\mathsf{y} = \mathsf{Yes}$. This private preference y is masked with probability $1 - \mathsf{p}$. Then a fair coin is tossed to output label Yes with likelihood $\frac{1-\mathsf{p}}{2}$.

Referencing figure 3.4, the conditional probabilities are:

$$\mathfrak{p}(\tilde{\mathsf{y}} = \mathsf{Yes} \mid \mathsf{y} = \mathsf{Yes}) = \mathsf{p} + \frac{1}{2}(1-\mathsf{p}) \qquad \mathfrak{p}(\tilde{\mathsf{y}} = \mathsf{Yes} \mid \mathsf{y} = \mathsf{No}) = \frac{1}{2}(1-\mathsf{p})$$
$$\mathfrak{p}(\tilde{\mathsf{y}} = \mathsf{No} \mid \mathsf{y} = \mathsf{Yes}) = \frac{1}{2}(1-\mathsf{p}) \qquad \mathfrak{p}(\tilde{\mathsf{y}} = \mathsf{No} \mid \mathsf{y} = \mathsf{No}) = \mathsf{p} + \frac{1}{2}(1-\mathsf{p}). \tag{3.2}$$

Here y_k *is the private true label that* Wit_k *give to the query-response pair* $(\mathbf{x}_{\tau-1}, \mathbf{x}_\tau)$*, and* $\tilde{\mathsf{y}}$ *is the published label scrambled by* FeaAudit*. Critically, observe that without knowing* r_k*, we have* $\mathfrak{p}(\tilde{\mathsf{y}} = \mathsf{Yes}) = \mathfrak{p}(\tilde{\mathsf{y}} = \mathsf{No})$ *under the privacy mask of* coin_p*. So one cannot determine the* Wit*ness's true preferences.*

Observation 3.3.2. *(The* Cohort *is Exposed) This follows immediately from:*

$$\frac{\mathfrak{p}(\tilde{\mathsf{y}} = \mathsf{Yes} \mid \mathsf{y} = \mathsf{Yes})}{\mathfrak{p}(\tilde{\mathsf{y}} = \mathsf{Yes} \mid \mathsf{y} = \mathsf{No})} = \frac{\frac{\mathsf{p}+1}{2}}{\frac{1-\mathsf{p}}{2}} = \frac{\mathsf{p}+1}{1-\mathsf{p}}.$$

So FeaAudit *is* $\ln\left(\frac{\mathsf{p}+1}{1-\mathsf{p}}, 0\right)$*-differentially private as seen in definition B.1.2.*

Although the spectators cannot determine any Witness's private preference on an individual basis, the population preference of any Cohort may still determined. Since we have $\mathfrak{p}(\tilde{y} = \mathsf{Yes} \mid y = \mathsf{Yes}) > \mathfrak{p}(\tilde{y} = \mathsf{Yes} \mid y = \mathsf{No})$ and so on, as seen in eqn 3.2. The next proposition make this statement precise.

Proposition 3.3.3. *(*Cohort *labels are "sufficiently close") to the true label in every iteration of* FeaAudit*, provided the* PA*-set* $\mathsf{coin}_{\mathsf{p}}$ *is defined so that:*

$$\mathsf{p} \geq 1 - \frac{d}{\mathsf{N}}, \tag{3.3}$$

for some PA *defined* $d \in \{0, 1, 2, \ldots\}$.

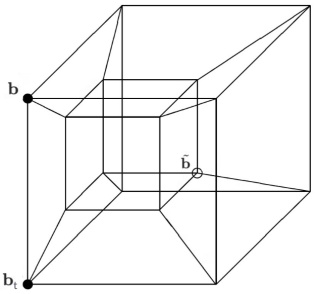


Figure 3.5: The set of all N-bit strings forms a hypercube with 2^{N} vertices, whereby we traverse any edge by flipping just one bit. Lines 14-22 of FeaAudit then correspond to a random walk on this hypercube. Starting at the true vertex $\mathbf{b}$, it either flips the t-th bit to reach vertex $\mathbf{b}_{\mathsf{t}}$, or do not flip any bits and stay at $\mathbf{b}$. After N possible flips, we arrive at $\tilde{\mathbf{b}}$.

Proof. Immediately follows from law of large numbers. C.5 goes through the motions. ■

Observation 3.3.4. *(The Prince is Protected while the Eunuch is Exposed). At any stage* t*, the* PA *reveals his judgment of some* Wit_k^m*'s evaluation of the model by passing a bent* $\mathsf{coin}_{\mathsf{q}}^{\mathsf{t}}$ *with hidden bias* q_{t}*. Now referencing line* 28-31 *of* FeaAudit*, the two tosses of the* VA*'s bent* $\mathsf{coin}_{\mathsf{q}}^{\mathsf{t}}$ *creates a new bent coin. Given* $\mathfrak{p}(\mathsf{coin}_{\mathsf{q}}^{\mathsf{t}} = \mathsf{Head}) = \mathsf{q}_{\mathsf{t}}$*, here are three possible outcomes to* VA*'s random experiment with their associated likelihoods:*

Head, Head	$: \mathsf{q}_{\mathsf{t}}^2$
Head, Tail	$: \mathsf{q}_{\mathsf{t}}(1 - \mathsf{q}_{\mathsf{t}})$
Tail	$: 1 - \mathsf{q}_{\mathsf{t}}$.

They correspond to how a rational VA *behaves as he selfishly* ***reject*** *the* Wit*nesses and appropriate their wealth. For reasons that will be clear later, we call this the Non-Veumann method, which takes a bent coin and bend its bias towards your favor. The bias of this derived bent coin is then:*

$$\mathfrak{p}(\mathsf{Head}) = \mathsf{q}_{\mathsf{t}}^2 \qquad \mathfrak{p}(\mathsf{Tail}) = \mathsf{q}_{\mathsf{t}}(1 - \mathsf{q}_{\mathsf{t}}) + 1 - \mathsf{q}_{\mathsf{t}} = 1 - \mathsf{q}_{\mathsf{t}}^2. \tag{3.4}$$

In concrete terms, observe by definition the PA*'s bent coin* $\mathsf{coin}_{\mathsf{q}}^{\mathsf{t}}$ *must have bias between* $\mathsf{q}(\frac{1}{3}, \frac{2}{3})$*, so that:*

$$0.556 \leq \mathfrak{p}(\mathsf{Tail}) \leq 0.889.$$

The odds are stacked in favor of the VA *once he is selected onto the eunuch position. The expected number of stages* t^* *any* VA_k *can survive before seeing the elimination sequence of* Head, Head *is bounded by:*

$$\frac{1}{\mathsf{q}_u^2} \leq \mathsf{t}^* \leq \frac{1}{\mathsf{q}_l^2} \implies \mathsf{t}^* \in [2.25, 9]. \tag{3.5}$$

So the eunuch's expected tenure at court when flipping the coin*s to survive exceeds* 2 *stages. If the eunuch is honest and only tosses each* $\mathsf{coin}_{\mathsf{q}}^{\mathsf{t}}$ *once, then his expected survival time is:*

$$\frac{1}{\mathsf{q}_u} \leq \mathsf{t}^* \leq \frac{1}{\mathsf{q}_l} \implies \mathsf{t}^* \in [1.5, 3].$$

Whenever Wit*nesses observe some eunuch* VA_k *survives more than* 3 *stages of the game, then they have reason to suspect* VA_k *is tossing in a self-interest manner. In conclusion,* Wit*nesses cannot determine if the* PA *is truly displeased with their response, or if it is the self-serving* VA*-eunuch tossing and raking. Meanwhile the eunuch is exposed for his greed, not his true belief as to whether a label is correct or not.*

We built FeaAudit with the intention of protecting the prince PA on a per-Witness basis, so the VA filter offers some protection against the mob. Additionally, the bias of the bent $\mathsf{coin}_{\mathsf{q}}^{\mathsf{t}}$ is q_{t} indexed by t, so it is different on every stage t. However, if the PA demonstrates consistent bias towards a specific direction, then over the long run the Witnesses may be able to determine the PA's secret preference. Once the PA's preference is known, the Witnesses may cheat by testifying in a manner that appeals to the PA's prejudices. Remark 3.4.4 addresses this problem by running many FeaAudit instances in parallel.

Remark 3.3.5. (*Alternate randomization technique to protect the* PA) We can protect the preference of the PA further if the eunuch VA_k uses the "Von Neumann method" [Mit13]:

3.3.5.1. toss $\mathsf{coin}_{\mathsf{q}}^{\mathsf{t}}$ twice, if they show Head, Tail or Tail, Head, then disclose the 1st toss;
3.3.5.2. if the two tosses come up Head, Head or Tail, Tail, then toss the coin again twice.
3.3.5.3. This process repeats until the VA_k sees two tosses of different faces.

This sampling regime creates a fair coin out of a bent coin of any bias q, and thus protects the prince PA's true preference absolutely. However, this may not be ideal as:

3.3.5.1. the prince or PA may want to shape public opinion subtly to his own.
3.3.5.2. It defeats the purpose of having the PA judge the testimony at all.
3.3.5.3. It is too laborious for the VA_k to do.

In general, the randomization technique 28-32 and 14-22 may be an external process that is composed with the main FeaAudit protocol, just as QAudit-machine-learning subroutines can be composed with the main QAudit body.

Now we discuss how players profit from FeaAudit:

1. *The prince* PA: is the most privileged position. His preferences are hidden and he never loses money. Instead he collects $\frac{1}{10}$ of every Witness's or eunuch's funds regardless how the coin tosses land. The PA is in effect taxing the users by running a *fairness farm*, and in doing so helping Metheus tax the users as well.

2. *The* Wit*ness*: the witness may earn money passively simply for sitting in one of the Cohorts, and is showered with some fraction of each rejected Witness or eunuch's fund. We call this behavior *fairness farming*, and it is the easiest way to enter the Metheus ecosystem. However, once the Witness is called to judge a query-response pair, the odds are less favorable as seen in observation 3.3.4. Nevertheless, it is possible for Witnesses to sit in a Cohort and never be called upon to testify before a query. In fact, FeaAudit rules demands it as by definition, there exists one Cohort whose members have not been all eliminated at end of the game. Knowing this, Witnesses may cheat by shuffling themselves into the largest Cohort. This problem can be easily remedied by capping the size of every Cohort.

3. *The eunuch* VA_k: although the eunuch is exposed, he is compensated with better odds. *In expectation* w.r.t q and ξ, the VA_k may survive long enough to exit the eunuch position with more wealth than he enters with. Specifically, we know from 3.3.4 the survival rate is $0.556 \leq \mathfrak{p}(\mathsf{Tail}) \leq 0.889$. The next proposition determines under what condition the VA_k may die richer.

Proposition 3.3.6. *(Eunuchs who die old, die rich). That is every eunuch* VA_k *can expect to die wealthier than when he is selected into the position, if* FeaAudit *set wealth seizure parameter so that:*

$$0.384 < \xi < 0.80. \tag{3.6}$$

Proof. See C.5. ■

Protecting the eunuch's interest is central to the FeaAudit incentive structure, as it prompts the eunuch to *always play selfishly*. In doing so, they can be relied upon to hide the prince's true intentions. The more self-serving the eunuch, the more protected the prince. This is why eunuchs exist. In fact the eunuch's singular greed for $\mathfrak{miat}$ also protects *him*, because the Witnesses can be rest assured that the eunuch believes in nothing - a fact that has many advantages. The next section extends FeaAudit so that Witnesses can also protect their $\mathfrak{miat}$-downside.

3.4 Tokenizing FeaAudit

A refinement on the baseline game is to tokenize the FeaAudit fairness oracle. Specifically, each Witnesses may sell a *fair and equal annotation token*, or $\mathfrak{feat}$. Each $\mathfrak{feat}_\mathsf{k}$ is a measure of Wit_k's earning potential in the FeaAudit game, and is essentially a claim on some portion of the total pot of $\mathfrak{miat}$ in any FeaAudit game. This $\mathfrak{feat}$ is marketed as follows:

1. a Witness Wit_k enters an instance of the FeaAudit game, and deposit the minimum $\mathfrak{miat}$ balance $\mathfrak{v}$. Denote this initial wealth with:

   $$\mathfrak{v}_\mathsf{k}^o = \mathfrak{v}. \tag{3.7}$$

   Conceptually, Wit_k just purchased a $\mathfrak{feat}_\mathsf{k}$ option with initial payment $\mathfrak{v}$.

2. At any stage t, a Wit_k may mint *one* edition of $\mathfrak{feat}_\mathsf{k}$, and sells it to an $\mathsf{Auditor}_\mathsf{i}$ for some price. In doing so, the $\mathsf{Auditor}_\mathsf{i}$ has assumed the entire up and downsides of Wit_k. Specifically:

   a) when the Wit_k is airdropped $\mathfrak{miat}$ from rejected eunuchs and other Witnesses as seen in line 38 and 46 of FeaAudit, this $\mathfrak{miat}$ is owed to $\mathsf{Auditor}_\mathsf{i}$. We can denote the deposited balance at this stage t with:

   $$\mathfrak{v}_\mathsf{k}^\mathsf{t} = \mathfrak{v} + \mathsf{t}\Delta\mathfrak{v}, \tag{3.8}$$

   where the constant $\Delta\mathfrak{v}$ is the airdropped $\mathfrak{miat}$ from rejected Witnesses. This $\mathsf{t}\Delta\mathfrak{v}$ is the "leverage" gained by simply sitting in the FeaAudit market.

   b) When Wit_k is called to testify the fairness of some query response pair $(\mathbf{x}_{\tau-1}, \mathbf{x}_\tau)$ and becomes a eunuch VA_k, then the entire upside of VA_k's earnings is also owed to $\mathsf{Auditor}_\mathsf{i}$. We can denote VA_k's current wealth at any stage t with:

   $$\mathfrak{v}_\mathsf{k}^\mathsf{t} = \mathfrak{v} + (\mathsf{t}^o - 1)\Delta\mathfrak{v} + \frac{\xi}{2}\sum_{\mathsf{t}^o}^{\mathsf{t}} \mathfrak{v}_\mathsf{i}, \quad \text{where } \mathsf{i} \in \{1, \ldots\}/\mathsf{k}. \tag{3.9}$$

   Here the index t^o is when VA_k became a eunuch, and t is the current game stage so that $\mathsf{t}^o \leq \mathsf{t}$. The Witness is conceptually purchasing his seat as a eunuch with his own money $\mathfrak{v}$, and leverage $(\mathsf{t}^o - 1)\Delta\mathfrak{v}$.

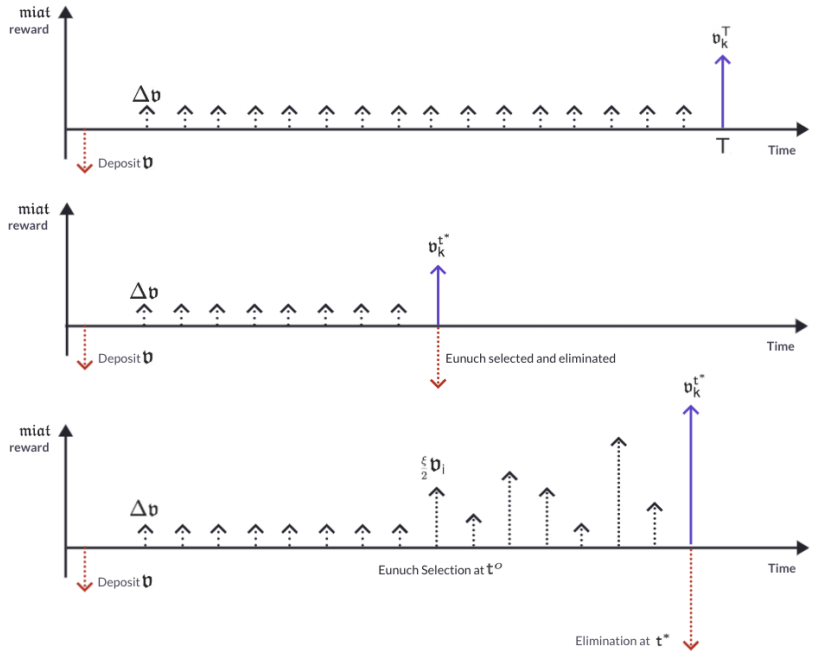


Figure 3.6: Three payout scenarios for FeaAudit. The top graph corresponds to eqn 3.10, where the Witness Wit_k deposits $\mathfrak{v}$, sits in the winning Cohort, collect airdrops from other eliminated Witnesses. Notably he is never eliminated himself, so the entire amount $\mathfrak{v}_k^T$ is given to the $\mathfrak{feat}_k$ holder. In the middle graph for 3.11, the Witness is chosen to testify, and is immediately eliminated by the sitting eunuch. So the $\mathfrak{feat}_k$ holder receives $(1-\xi)\mathfrak{v}_k^{t^*}$. In the bottom graph, the Witness becomes the eunuch, and survives many stages as a eunuch. He earns a variable amount of $\frac{\xi}{2}\mathfrak{v}$ between t^o and t^*, and is eliminated at t^*. So he returns $(1-\xi)\mathfrak{v}_k^{t^*}$ to $\mathfrak{feat}_k$-holder.

3. When this Wit_k exists the game at stage t^*, whether as a rejected Witness or a rejected eunuch, the $\xi\mathfrak{v}_k^{t^*}$ downside is paid from the $\mathfrak{miat}$ that is otherwise owed to the $\mathfrak{feat}_k$ owner, namely $\mathsf{Auditor}_i$. Each Wit_k's terminal wealth at elimination is:

$$\mathfrak{v}_k^T = \mathfrak{v} + T\Delta\mathfrak{v} \text{ if does not die,} \tag{3.10}$$

$$\mathfrak{v}_k^{t^*} = \mathfrak{v} + (t^* - 1)\Delta\mathfrak{v} \text{ if dies as Witness,} \tag{3.11}$$

$$\mathfrak{v}_k^{t^*} = \underbrace{\mathfrak{v} + (t^o - 1)\Delta\mathfrak{v}}_{\text{prefix}} + \underbrace{\frac{\xi}{2}\sum_{t=t^o}^{t^*}\mathfrak{v}_i}_{\text{suffix}} \text{ if dies as eunuch.} \tag{3.12}$$

So the $\mathsf{Auditor}_i$'s "take-home" wealth from $\mathfrak{feat}_k$ option is $(1-\xi)\mathfrak{v}_k^{t^*}$.

Now consider how each side would profit:

1. the $\mathsf{Auditor}_i$ only pays once: when he buys $\mathfrak{feat}_k$. He profits if the price of $\mathfrak{feat}_k$ is less than $(1-\xi)\mathfrak{v}_k^{t^*}$, he loses $\mathfrak{miat}$ otherwise. So Auditors should buy $\mathfrak{feat}_k$ from Wit_k if he believes this Witness has a "good chance" of either 1) surviving to the end without being called to testify. Or 2) surviving many rounds as a eunuch, which entails two factors:

   a) Wit_k is drawn from a Cohort whose bias is similar to that of the prince PA.

b) When this $\mathsf{Wit_k}$ becomes the eunuch $\mathsf{VA_k}$ in the future, FeaAudit will draw a runs of Witnesses s.t. 1) their judgments either displeases the PA, and/or 2) the $\mathsf{VA_k}$ can continuously flip $\mathsf{coin}_\mathsf{q}^\mathsf{t}$'s where he does not see the elimination pair of Head, Head. Critically, recall the likelihood of surviving each stage is greater than 55% regardless of how the PA answers.

In sum, Auditors who purchase $\mathfrak{feat}_\mathsf{k}$ are fundamentally speculating on the preference of the prince PA over fairness judgments.

2. The $\mathsf{Wit_k}$ pays $\mathfrak{v}$ upon entry, and his wealth increase on every stage by some $\Delta\mathfrak{v}$ simply for sitting in a Cohort. So he may decide at some stage t to sell $\mathfrak{feat}_\mathsf{k}$ and lock in his gains. There are three possible price points he may accept:

   a) Strictly speaking, $\mathsf{Wit_k}$ profits if the price of $\mathfrak{feat}_\mathsf{k}$ is greater than $\mathfrak{v}$ - the price he paid to enter the game.

   b) Alternatively he may be greedy, and set the threshold at $\mathfrak{v} + (\mathsf{t}-1)\Delta\mathfrak{v}$.

   c) The most conservative point is $(1-\xi)(\mathfrak{v} + (\mathsf{t}-1)\Delta\mathfrak{v})$, that is how much $\mathsf{Wit_k}$ would be left with if he is called upon to testify, and immediately is immediately rejected. This is effectively a fire sale of his position.

We assume the Witness is irrational, so any price point above is possible. However, the real question is what is the fair market value of this $\mathfrak{feat}_\mathsf{k}$ from the Auditor's perspective. We can express a lower-bound the the value of $\mathfrak{v}_\mathsf{k}^{\mathsf{t}^*}$ of eqn 3.12 as follows. Given some $\mathsf{Wit_k}$ is selling $\mathfrak{feat}_\mathsf{k}$ at stage t, then we know the prefix is bounded by:

$$\begin{aligned} \alpha\mathfrak{v} + (\mathsf{t}-1)\Delta\mathfrak{v} \leq \mathsf{prefix} \leq \alpha\mathfrak{v} + \mathsf{T}\Delta\mathfrak{v} \\ \text{where } \Delta\mathfrak{v} \geq 0,\ \alpha \geq 1. \end{aligned} \tag{3.13}$$

Here T denotes the end of the FeaAudit game as defined by the while loop, while $\alpha\mathfrak{v}$ denotes the actual amount the Witness spent to enter the market. So in the worst case, $\mathsf{Wit_k}$ is selected for testimony in the next stage, while in the best case, the $\mathsf{Wit_k}$ survives until the final stage T. If the Auditor does not expect the Witness to be chosen, then his price quote for $\mathfrak{feat}_\mathsf{k}$ should lie within $(1-\xi)\mathsf{prefix}$.

Next we assume the Witness will be chosen as a eunuch $\mathsf{VA_k}$ at some stage, and wlog call this stage 1. Then we can bound the the suffix of eqn 3.12, that is $\mathsf{VA_k}$'s earnings over any game interval.

Proposition 3.4.1. *(The VA_k's expected earning) over any interval $[1, \mathsf{t}_u^*]$ is bounded with:*

$$2\beta\mathfrak{v}\xi \sum_{\mathsf{t}^*=0}^{\mathsf{t}_u^*} \left(\mathsf{t}^*+2\right)\frac{5^{\mathsf{t}^*}}{9^{\mathsf{t}^*+1}} \ \leq\ \mathbf{E}\Big[\mathsf{VA}_k\text{'s income on } [1,\mathsf{t}_u^*]\Big] \ \leq\ \frac{\beta\mathfrak{v}\xi}{2} \sum_{\mathsf{t}^*=0}^{\mathsf{t}_u^*} \left(\mathsf{t}^*+2\right)\frac{8^{\mathsf{t}^*}}{9^{\mathsf{t}^*+1}}. \tag{3.14}$$

Where:

3.4.1.1. *$\beta\mathfrak{v}$ with $\beta \geq 1$ denotes the average wallet balance of* Wit*nesses;*
3.4.1.2. *stage 1 is the moment VA_k becomes a eunuch;*
3.4.1.3. *t_u^* is any value less than the upper bound on the while loop of* FeaAudit*;*
3.4.1.4. *$\mathsf{t}^* \in [1, \mathsf{t}_u^*]$ denotes the stage where VA_k is eliminated.*
3.4.1.5. *The expectation is taken w.r.t the $\mathsf{coin}_\mathsf{q}^\mathsf{t}$ tosses of line* 29-31*.*

See fig 3.7 for game tree used to express the bound in eqn 3.14.

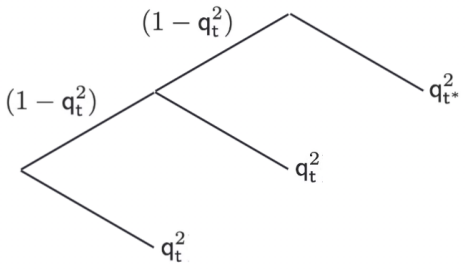


Figure 3.7: At every stage $\mathfrak{t}$, the eunuch VA_k dies with likelihood $\mathfrak{q}_{\mathfrak{t}}^2$, and continues to next stage $\mathfrak{t}+1$ with likelihood $1-\mathfrak{q}_{\mathfrak{t}}^2$. At the terminal stage $\mathfrak{t}^*$, VA_k dies with likelihood $\mathfrak{q}_{\mathfrak{t}^*}^2$.

Proof. See C.5. ■

Now we can express the fair price range of $\mathfrak{v}_\mathsf{k}^{\mathfrak{t}_u^*}$.

1. Using the lower bound in eqn 3.13 we state the range of expected values of $\mathfrak{v}_\mathsf{k}^{\mathfrak{t}_u^*}$, in the event where the Auditor expects the Witness to be chosen as a eunuch at stage $\mathfrak{t}+1$:

$$\alpha\mathfrak{v}+\mathfrak{t}\Delta\mathfrak{v}+2\mathfrak{v}\beta\xi\sum_{\mathfrak{t}^*=\mathfrak{t}}^{\mathfrak{t}_u^*}\Big(\mathfrak{t}^*+2\Big)\frac{5^{\mathfrak{t}^*}}{9^{\mathfrak{t}^*+1}}\ \le\ \mathbf{E}\big[\mathfrak{v}_\mathsf{k}^{\mathfrak{t}_u^*}\big]\ \le\ \alpha\mathfrak{v}+\mathfrak{t}\Delta\mathfrak{v}+\frac{\beta\mathfrak{v}\xi}{2}\sum_{\mathfrak{t}^*=\mathfrak{t}}^{\mathfrak{t}_u^*}\Big(\mathfrak{t}^*+2\Big)\frac{8^{\mathfrak{t}^*}}{9^{\mathfrak{t}^*+1}}. \tag{3.15}$$

   The Auditor may now select suitable α, β, ξ, and $\mathfrak{t}_u^*$ to determine where the asking price of $\mathfrak{feat}_\mathsf{k}$ rests w.r.t this range.

2. We can redefine the lower bound as the event where the Witness is called to testify on stage $\mathfrak{t}+1$, and immediately eliminated:

$$\alpha\mathfrak{v}+\mathfrak{t}\Delta\mathfrak{v}\ \le\ \mathbf{E}\big[\mathfrak{v}_\mathsf{k}^{\mathfrak{t}_u^*}\big]\ \le\ \alpha\mathfrak{v}+\mathfrak{t}\Delta\mathfrak{v}+\frac{\beta\mathfrak{v}\xi}{2}\sum_{\mathfrak{t}^*=\mathfrak{t}}^{\mathfrak{t}_u^*}\Big(\mathfrak{t}^*+2\Big)\frac{8^{\mathfrak{t}^*}}{9^{\mathfrak{t}^*+1}}. \tag{3.16}$$

Depending on the constants, he may pay for any price within this $(1-\xi)\mathbf{E}[\mathfrak{v}_\mathsf{k}^{\mathfrak{t}_u^*}]$ interval. It goes without saying that the price in eqn 3.16 should be updated on every stage $\mathfrak{t}$ to reflect the change in $\mathfrak{t}\Delta\mathfrak{v}$. Alternatively, one may set $\Delta\mathfrak{v}$ and $\mathfrak{t}^o$ as random variables and model this expectation.

Remark 3.4.2. (*How* $\mathfrak{feat}_k$ *is Marketed in Practice*). There are two ways to sell $\mathfrak{feat}_\mathsf{k}$: push and pull. The "pull" model is presented above whereby Witnesses initiate the sale and pull Auditors into the market. In the "push" model which is more natural to how DeFi operates, Auditors initiate the sale by buying say 100 $\mathfrak{feat}$s at minimum price $\mathfrak{v}$, and staking 100 Witnesses into the game. In this case, the Auditors hold a diverse group of Witnesses, and hope at least a few will make it as eunuchs. Meanwhile Witnesses play for free. Moreover, the Auditors and Witnesses may strike some standardized term where by:

3.4.2.1. The Auditors receive the taxed portion of their principle in $(1-\xi)\mathfrak{v}$, along with the taxed portion of the eunuch's upside upper-bounded by $(1-\xi)\frac{\beta\mathfrak{v}\xi}{2}\sum_{\mathfrak{t}^*=\mathfrak{t}}^{\mathfrak{t}_u^*}\Big(\mathfrak{t}^*+2\Big)\frac{8^{\mathfrak{t}^*}}{9^{\mathfrak{t}^*+1}}$.

3.4.2.2. The Witnesses receive the after-tax portion of airdrops from other rejected Witnesses in $(1-\xi)\mathfrak{t}^*\Delta\mathfrak{v}$.

Under these terms, the Witnesses only has upside measured in $\mathfrak{miat}$, no downside other than time lost. Alternatively, the Auditor may want some claw back provisions as downside protection if he stakes $\alpha\mathfrak{v}$, so Witnesses may receive less than $(1-\xi)\mathfrak{t}^*\Delta\mathfrak{v}$ in payment. Exact details will be at the Auditors' discretion.

Remark 3.4.3. (FeaAudit *conversion loop*)

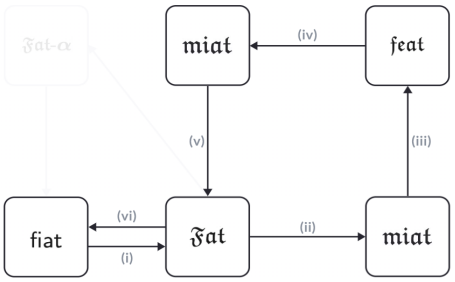


Figure 3.8: When Witnesses or Auditors buy into the FeaAudit annotation game by purchasing $\mathfrak{feat}$, they undergo the same conversion cycle as fig 2.8, with the commensurate dilution in upside that be recovered by holding $\mathfrak{Fat}$ tokens. Thus FeaAudit is both a $\mathfrak{miat}$ token sink and a token source. See fig 4.6 for more information.

Remark 3.4.4. (*Scaling* FeaAudit). In the basic FeaAudit formulation, there is a bottleneck in decision making, as the prince PA must judge every testimony from every Witness. Similar to how QAudit may be chain in series, FeaAudit may be chain in parallel. Whereby i.e. four princes may run four instances of FeaAudit together, and Witnesses are routed to a random prince in every stage t. Now each prince only has to judge $\frac{1}{4}$ of the Witnesses pool, naturally the "tax revenue" is also split by commensurate amount. The Witness's testimony is judged by his assigned prince PA, and may become a eunuch to this very PA. Running many instances of FeaAudit in parallel protects the prince even more, as routing the Witness towards any prince occurs randomly.

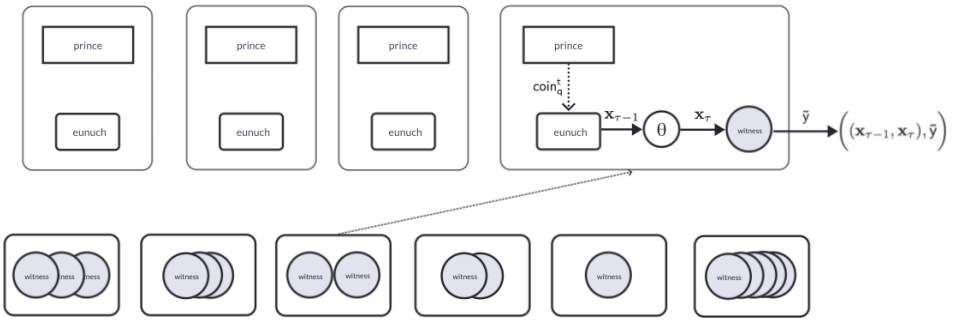


Figure 3.9: FeaAudit with four princes, and their respective eunuchs. Witnesses are routed to a random princes for testimony.

3.5 Conclusion

This chapter specifies the fair and equal annotation audit or FeaAudit. The protocol specifies a family of fairness oracles, or annotator services whereby Witnesses are called forth to testify the social compliance of a generative model. The prince or PA then judges the response, and passes his opinion in secret to the eunuch or the VA. Together they make a decision on whether to reward the Witness or reject him. Critically, both the Witness's labels and the prince's judgments are masked by coin tosses. This gives each party plausible deniability. The coin tosses are structured such that downstream algorithms may nonetheless determine population level preferences over Cohorts of Witnesses, perhaps segregated by race, gender, income etc. Thus FeaAudit streams useful labeled data, while satisfying the promises of

differential privacy for the Witnesses. Moreover, Witnesses may sell their future earnings with fairness tokens or $\mathfrak{feat}$. Unlike other DeFi coins whose value is "YOLO or rug," $\mathfrak{feat}$ can be priced via *technical valuation methods.* Its price is a function of the market structure and market constants of FeaAudit, these are known to the buyers/Auditors, and can be determined by eqn 3.16.

Here we used fairness annotation as a framing device to craft a technical narrative. However structurally there is nothing unique about this class of labels. FeaAudit can be easily adapted to any arbitrary labeling task. For example, instead of judging a generative model, a variation of FeaAudit may judge a data set instead. In this case, the Witnesses may be called forth to outline pedestrians in some video. The PA and VA then work in concert to determine if the Witness's labels are "good" by some subjective measure. In fact, a cluster of FeaAudit instances are annotator farms in the Metheus ecosystem, and allow us to generate *labeled* data that improve the quality of models trained in the QAudit instances. Unlike Amazon's Mechanical Turks or other services of similar flavor, the annotators are not paid to label. Instead they are *paying* to label data. From an in-ecosystem currency stand point, the QAudit markets are the $\mathfrak{miat}$ sources, while FeaAudit instances are $\mathfrak{miat}$ sinks. They work in concert to modulate the liquid supply of $\mathfrak{miat}$ across Metheus markets.

PART II

Stablecoin Protocol

CHAPTER 4

FAT and Stable

4.1 Introduction

The chapter completes the Metheus economy with the Fair, Accountable, and Transparent fiat-pegged stablecoin, or the $\mathfrak{Fat}$ stablecoin. $\mathfrak{Fat}$ will be the DeFi currency of interest for most retail users of Metheus. In the previous chapters we saw $\mathfrak{Fat}$ is exchanged for $\mathfrak{miat}$ along well-defined exchange rates, and $\mathfrak{miat}$ is needed in all the Metheus-related market operations. Thus, the growth of $\mathfrak{Fat}$ supply and its speculative upside is tied to the overall health of the Metheus economy. The fundamentals of which are:

1. the quality and volume of AI models or training data procured by QAudit markets;
2. the quantity of fairness annotations and labeled data from FeaAudit oracles;
3. the overall rate of transactions relating to $\mathfrak{miat}$-denominated wagers.

Thus $\mathfrak{Fat}$ is the first **entropy-stablecoin** tied to a bundle of essential commodities in the information economy. This is conceptually analogous to the petrodollar, also tied to a critical commodity of the modern world. Unlike the petrodollar however, Metheus **makes wagers, not war.** Because $\mathfrak{Fat}$ is based on a real economy that generates cash flows by pushing in-demand produce, it can be designed with the holder's long-term safety in mind. This means $\mathfrak{Fat}$ is fully fiat-collateralized, and features:

1. no margin calls,
2. no forced liquidations,
3. no funny business of any kind.

In sum: $\mathfrak{Fat}$ is real money, not funny money. The rest of chapter proceeds as follows:

1. section 4.2 outlines the $\mathfrak{Fat}$ issuance and redemption mechanism. In particular:
   a) primary mint of $\mathfrak{Fat}$ on the Metheus sidechain;
   b) and secondary issuance of $\mathfrak{Fat}$-α and $\mathfrak{Fat}$-β on the Ethereum mainnet.
2. Section 4.3 expresses the exact relationship between $\mathfrak{Fat}$ and $\mathfrak{miat}$ on a exchange-curve that is piece-wise smooth. And completes the technical narrative of how fluctuations in their relative supply ensure $\mathfrak{Fat}$ has dividend paying value.
3. Section 4.4 develops a remarkably simple valuation methods for the $\mathfrak{Fat}$ stablecoin.

4.2 FAT Issuance and Redemption

This sections shows why $\mathfrak{Fat}$ issuance is the *raison d'être* for the Metheus sidechain.

FAT Retail Issuance

Figure 4.1 diagrams the issuance process, which originates from an off-chain $\mathfrak{Fat}$ facility and proceeds as follows:

Figure 4.1: Metheus initiates the deposit process by placing some κ fraction of 1 fiat into the pre-designated *off-chain* $\mathfrak{Fat}$ facility. Then the creditor deposits the remaining $1-\kappa$ fiat to make 1 fully-collateralized $\mathfrak{Fat}$ stablecoin on the Metheus sidechain. This $\mathfrak{Fat}$ receives pro-rata share of $\mathfrak{miat}$ inflation dividend pool until some predesignated expiration date, at which point the principle of 1 fiat can be withdrawn by $\mathfrak{Fat}$ holder.

1. Metheus initiates the origination process by depositing κ fraction of 1 fiat into the $\mathfrak{Fat}$ facility, where $0 < \kappa \leq \frac{1}{20}$. This initial Metheus outlay comes from Metheus's operating margins, and is booked as the "cost of deposit acquisition."

2. A creditor approaches Metheus and deposits $1-\kappa$ in equivalently denominated fiat into the same $\mathfrak{Fat}$ facility. The creditor receives 1 $\mathfrak{Fat}$ stablecoin minted onto the Metheus sidechain. Observe the total $\mathfrak{Fat}$ supply in the Metheus economy now increases by 1, and there is exactly $1-\kappa+\kappa = 1$ fiat in the $\mathfrak{Fat}$ facility corresponding to this stablecoin. It is fully collateralized. This $\mathfrak{Fat}$ can be traded immediately on the sidechain. Now examine this arrangement from the two party's perspective:

   a) (creditor's perspective): the fair value of this $\mathfrak{Fat}$ is 1 fiat, however he only deposited $1-\kappa$. So his immediate profit for selling $\mathfrak{Fat}$ on the sidechain is up to κ. Alternatively he may hold and receive κ in profit at expiration.

   b) (Metheus's perspective): Metheus just borrowed fiat from the $\mathfrak{Fat}$-creditor at $\frac{1-\kappa}{\kappa}$ leverage. This fund can be reinvested in some low risk instruments to cover operating costs. See fig 4.2 for system value flow diagram.

   In this issuance arrangement, the rate of growth in $\mathfrak{Fat}$ is tied to growth in models, data, and annotation in the Metheus ecosystem. So $\mathfrak{Fat}$ supply is a levered measure of the system capacity to process information, hence the name entropy stablecoin. Finally, observe that since the $\mathfrak{Fat}$-creditor is interfacing with traditional financial (TradFi) systems at the point of issuance, KYC/AML is part of the purchasing process.

3. Alternatively, the creditor may "wrap" this sidechain $\mathfrak{Fat}$ stablecoin and move it onto Ethereum mainnet. $\mathfrak{Fat}$ follows the design pattern of [SCM21], and factors into two "red-black" primitives:

   a) ($\mathfrak{Fat}$-$\boldsymbol{\alpha}$ *inflation dividend token*): this is a claim on the inflation dividend pool. Inflation arise from the conversion process in fig 2.8 and fig 4.6. Where Auditors buy into QAudit/FeaAudit at the current fiat:$\mathfrak{Fat}$:$\mathfrak{miat}$ exchange rates, and cash out at higher exchange rates so that their winnings are diluted by increase in $\mathfrak{miat}$ supply. $\mathfrak{Fat}$-$\boldsymbol{\alpha}$ also have a expiration date, i.e. 6 month, until which time $\mathfrak{Fat}$-$\boldsymbol{\alpha}$ will accumulate its pro-rata share of inflation dividends. Each $\mathfrak{Fat}$-$\boldsymbol{\alpha}$ is owed:

$$\text{dividend owed per time period per } \mathfrak{Fat}\text{-}\boldsymbol{\alpha} = \frac{\text{fiat denominated inflation pool}}{\text{sply}(\mathfrak{Fat}\text{-}\boldsymbol{\alpha})}. \quad (4.1)$$

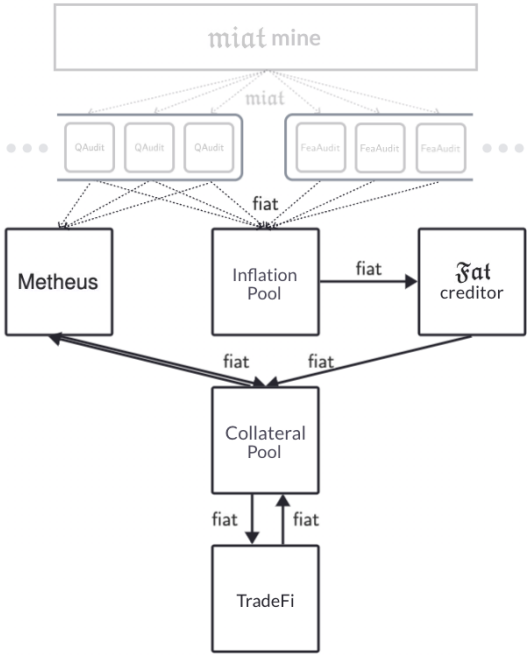


Figure 4.2: In the Metheus operating model, fiat dividend due to miat-expansion is placed into the *Inflation Pool*. Metheus and Fat-creditors then place fiat into a joint *Collateral Pool*. This pool invests in TradFi instruments. Returns from this investment are claimed by Metheus, while miat-inflation dividend are sent to Fat-creditors.

Where sply(Fat-α) refers to the total supply of *unexpired* outstanding Fat-α. Each time period may last i.e. 7 days, so that every Sunday the entire fiat-denominated-inflation pool is distributed to all the Fat-α holders. After the 6 months expiration date, the accumulation stops, and it is taken out of the payout pool, and the Fat-α supply decreases.

The owner of this expired Fat-α may then burn it at his own convenience. Note this implies Fat-α-supply may grow unbounded. This is OK as tranches of Fat-α will expire at their respective expiration date, so that the supply of un-expired Fat-α that partakes in the dividend pool does not grow at a faster rate than Fat is minted.

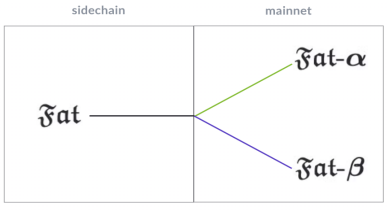


Figure 4.3: Fat issuance proceeds in two steps. 1) first a Fat stablecoin is minted on the Metheus sidechain, where it is fully-collateralized and pegged to 1 fiat of choice. 2) Then this Fat is split into Fat-β, which is redeemable for the 1 fiat underlying. And Fat-α, which claims the miat-inflation dividend upside. Both Fat-α and Fat-β are recorded on Ethereum mainnet, they have variable expiration rates and are tradable immediately.

b) (Fat-β *governance stablecoin*): this is a claim on the underlying 1 fiat deposit in the Fat facility. In particular, Fat-β has a some predefined lockup date, i.e. 6 month, after which it can be redeemed in full for the underlying 1 fiat. After redemption, both Fat-β on mainnet and one Fat on the Metheus sidechain are burnt. So Fat-β redemption decreases the supply of Fat in the Metheus sidechain, and decreases the exchange rate of miat:Fat in the QAudit/FeaAudit games. However the corresponding Fat-α need not be burnt. This implies the instanenous supply of Fat-α and Fat-β on mainnet differs.

Most importantly, Fat-β is a *governance token*: Fat-β holders can determine certain Metheus platform policy decisions, in particular:

i. miat:Fat exchange rate floor $\underline{e}$.
ii. miat:Fat exchange rate ceiling $\bar{e}$.

Fluctuations in exchange rate significantly affect Fat-α holders' dividend rewards:

i. expansion in exchange rate within any competition period $t = 1..T$ increases the amount of fiat-denominated wagers allocated to the inflation-dividend pool of fig 4.2. See fig 4.6 for conversion diagram.
ii. However, since Fat is pegged to 1 fiat of choice, the higher the miat:Fat exchange rate, the cheaper it is on a cash basis to wager in the Metheus markets. This reduces the wager pool on a fiat-basis, and reduces the fiat-dividend Fat-α holders receive from the inflation pool.
iii. Fat-β-holders must determine the exact balance. See fig 4.7 for visualization of how floor and ceilings affect exchange rate fluctuations.
iv. Because Fat-β-holders redeem collateral upon their own volition, unredeemed Fat-β stablecoins may sit on the book for an excessively long time. Thus the total supply of Fat may approach or exceed that of miat. Metheus use the communally-defined floor $\underline{e}$ to maintain the minimum exchange rate by minting 1 miat per Fat issued.

Finally, observe Fat-β is a fungible token, its value is 1 fiat modulo time value, and Metheus abide by one Fat-β one vote. However Fat-α is a nonfungible token, as its issuance and expiration date affects its value.

FAT Bulk Issuance

In an alternate Fat issuance structure, the creditor receives Fat-α only, and does not move the fiat principle as no Fat-β has changed hands. The creditor in-effect enters a swap structure, whereby he sets aside a tranche of capital, and trades interest rate accrued on capital (in red in fig 4.4) with variable miat-inflation dividend accrued to the the Fat-α he possesses.

Figure 4.4: In Fat bulk issuance, the creditor swaps fiat due to interest on principle he has set aside, against the fiat dividend from Fat-α he receives. Notably, Fat-α is exchanged at no cost, and there is no redemption process as the creditor does not have Fat-β on hand.

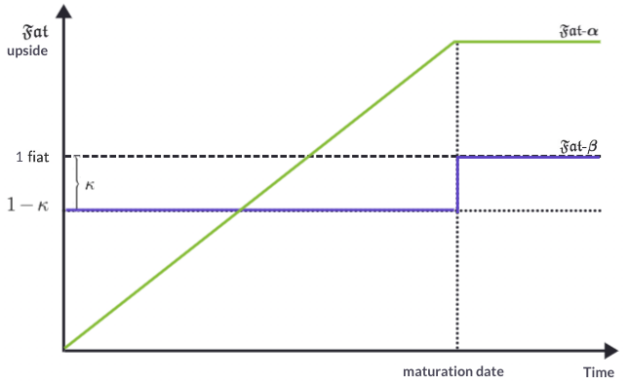


Figure 4.5: $\mathfrak{Fat}$ upside scenario, where green is $\mathfrak{Fat}$-$\boldsymbol{\alpha}$ and blue is $\mathfrak{Fat}$-$\boldsymbol{\beta}$ upside (best seen in color). In this case, the $\mathfrak{Fat}$-creditor buys $\mathfrak{Fat}$ in the sidechain for $1-\kappa$ fiat, and holds both primitives to expiration. He sees 1) a fixed upside of κ for $\mathfrak{Fat}$-$\boldsymbol{\beta}$ redemption from Metheus; and 2) receives rolling miat-inflation dividend from $\mathfrak{Fat}$-$\boldsymbol{\alpha}$ until expiration date. In this figure they expire on the same day. In reality this need not be the case.

As the section header suggests, $\mathfrak{Fat}$ bulk issuance is not a retail product. It is available only through private placement.

Qualitative Stakeholder Discussion

$\mathfrak{Fat}$ holders are exposed to the upside of Metheus economy in $\mathfrak{Fat}$-$\boldsymbol{\alpha}$, whilst carrying very little downside in holding $\mathfrak{Fat}$-$\boldsymbol{\beta}$. Moreover, by setting κ to be a function of interest rate in fiat-collateral of choice, the creditor's downside is at most zero. Once a creditor mints the $\mathfrak{Fat}$-$\boldsymbol{\alpha}$/$\mathfrak{Fat}$-$\boldsymbol{\beta}$ pair, he can sell both, hold both, or sell one whilst holding the other. Qualitatively speaking, these actions imply the following:

1. (Hold both $\mathfrak{Fat}$-$\boldsymbol{\alpha}$ and $\mathfrak{Fat}$-$\boldsymbol{\beta}$). In holding $\mathfrak{Fat}$-$\boldsymbol{\beta}$ until expiration, the creditor is delaying his profit of κ until redemption. In holding $\mathfrak{Fat}$-$\boldsymbol{\alpha}$, he is long the Metheus economy, and is rewarded with miat-inflation dividend until expiration.

2. (Hold $\mathfrak{Fat}$-$\boldsymbol{\alpha}$, sell $\mathfrak{Fat}$-$\boldsymbol{\beta}$). The creditor is long Metheus as before by holding $\mathfrak{Fat}$-$\boldsymbol{\alpha}$. However by selling $\mathfrak{Fat}$-$\boldsymbol{\beta}$ for up to 1 fiat, he is pocketing the $\leq \kappa$ profit now.

3. (Hold $\mathfrak{Fat}$-$\boldsymbol{\beta}$, sell $\mathfrak{Fat}$-$\boldsymbol{\alpha}$). The creditor delays the κ profit until expiration. But is pulling the upside from $\mathfrak{Fat}$-$\boldsymbol{\alpha}$ to the present day by selling. This could occur if he expects 1) QAudit/FeaAudit operations to enter a downturn, or 2) a deluge of $\mathfrak{Fat}$ buyers will enter the market and dilute the miat inflation pool.

4. (Sell both $\mathfrak{Fat}$-$\boldsymbol{\alpha}$ and $\mathfrak{Fat}$-$\boldsymbol{\beta}$). The creditor pockets the κ profit today, and also pulls the $\mathfrak{Fat}$-$\boldsymbol{\alpha}$ upside to the present day.

In sum, the creditor who initiates $\mathfrak{Fat}$ issuance has minimum upside of κ. Auditors should hold $\mathfrak{Fat}$-$\boldsymbol{\alpha}$ as a way to hedge inflation risk as they partake in the Metheus economy. Steps (vii) and (viii) in fig 4.6 are pivotal aspects of the overall Metheus tokenomic package. **In Metheus, auditors are creditors**. This aligns the incentives end-to-end. So that the Auditors not only rank the Agents in "good faith" in any particular QAudit market, they must also long the aggregate Metheus economy overall, if they wish to realize the full extent of their wagers.

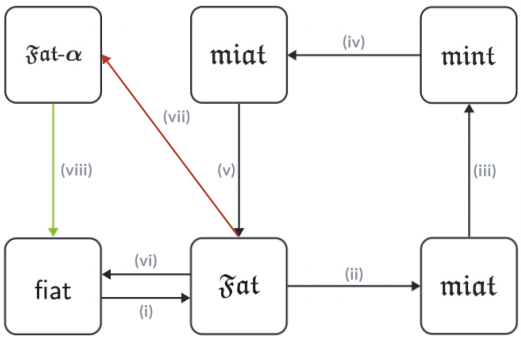


Figure 4.6: In this sequel to fig 2.8, the Auditor enters QAudit and sees his earnings diluted by miat inflation in steps (i) - (vi). In particular, value leakage occurs in step (v) when miat is converted back to Fat at the prevailing and higher exchange rate. This value is accrued to some tranche of Fat-α as dividend in step (vii). If the Auditor holds this tranche of Fat-α, then he sees his fiat upside returned to him in step (viii). Moreover, Auditors may also hold Fat-α to hedge their overall audit risk. So that even if they lose some particular stage t, they can nonetheless see some upside in inflation dividend.

4.3 FAT Exchange Rate

Since Metheus initiates the Fat issuance by depositing some amount κ fiat, the rate at which Fat is minted is fundamentally tied to the operating profit of Metheus. That is, the higher the QAudit/FeaAudit activity, the higher the miat supply to support the transaction volume, the greater the inflation and the higher the operating profit. The converse spiral is also true. And because Fat-α/Fat-β pairs mature in regular intervals, the supply of Fat is self-correcting. However, it is possible for the supply of Fat and miat to approach each other, or even for Fat to exceed miat supply. This is an adverse event that breaks the Metheus operating model.

Thus, Metheus features a *piece-wise exchange-rate bonding function* whereby within a certain exchange rate envelop, the miat:Fat exchange rate floats. However below this threshold $\frac{\text{miat}}{\text{Fat}} \leq \underline{\text{e}}$, one miat is minted for every Fat issued, so that the exchange rate invariant $\frac{\text{miat}}{\text{Fat}} > \underline{\text{e}}$ is maintained. Similarly, if the miat:Fat exchange rate drifts above the upper bound with $\frac{\text{miat}}{\text{Fat}} > \bar{\text{e}}$, then any miat in the global pool is burnt. This exchange rate floor is set by the holders of Fat-β, and constitutes the primary reason one would hold Fat-β instead of selling it to pocket the arbitrage profit. Now observe, Auditors that become Fat-creditors determine how much of their upside is taken by inflation. Naively, they would drive the floor $\underline{\text{e}}$ to be as low as possible. This would make sense if he:

1. expects to wager on the winning models in every stage in QAudit;
2. and/or expects to always stake the best eunuchs in every FeaAudit fairness pool.

However, if he expects to lose sometimes as FeaAudit and QAudit are designed to do, then he is better off setting a higher exchange rate floor. Critically: while inflation reduces the upside of Auditors, it also acts as *insurance* for the same Auditors *if* they hold Fat-α tokens, and receive inflation dividends from activities platform wide. Additionally, Auditors holding Fat-α tokens are also incentivized to bring other Auditors into the Metheus ecosystem, as their wagers also accrue to dividend pool. Thus Fat-α is both an Auditor-acquisition and Auditor-retention instrument.

The last point bears repetition, Fat-α tokens are *calls* for the underlying Metheus economy for retail buyers. However, if the Fat-α holder is also an Auditor, then he is buying *puts* on his own activity as an Auditor. Each Auditor must determine this exact balance of conflicting interests, the point of intersection is his ideal exchange rate floor $\underline{\text{e}}_{\text{m:f}}$. The Metheus governance process collates the private floors due to each creditor through a voting

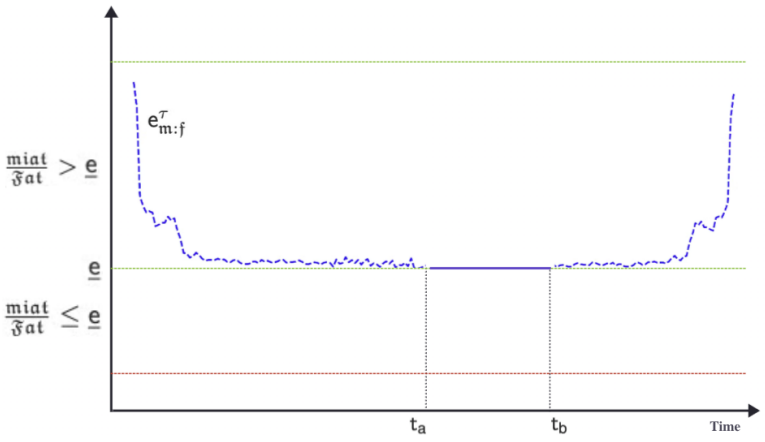


Figure 4.7: An exchange rate evolution scenario whereby miat:fiat exchange rate $e^{\tau}_{m:f}$ transitions from floating to fixed, and back to floating rate again. Between the time interval $[0, t_a]$, the miat:$\mathfrak{Fat}$ exchange rate is above floor $\underline{e}$. It precipitously falls towards value $\underline{e}$ as more $\mathfrak{Fat}$ are issued. On the interval $[t_a, t_b]$, the exchange rate is fixed at $\underline{e}$ as miat is mined in proportion to $\mathfrak{Fat}$ issuance. Beyond t_b, relative miat supply grows as QAudit/FeaAudit activity recovers, so the exchange rate is allowed to fluctuate per platform activity. Normally, this exchange rate floor $\underline{e}$ is set by $\mathfrak{Fat}$-β governance token holders. However, there is an exchange rate "hard-deck" defined by Metheus (red) that $\mathfrak{Fat}$-β holders cannot broach when voting. For example, this hard-deck may be 10:1, stating that the lowest exchange rate is 10 miat per $\mathfrak{Fat}$ stable token. Finally, there is also an exchange rate ceiling $\bar{e}$ (green) set by $\mathfrak{Fat}$-β holders, beyond which point miat tokens in the unclaimed global pool are burnt.

process, and sets the value of $e_{m:f}$ accordingly to provide collectively-defined level of $\mathfrak{Fat}$ social security.

4.4 FAT Valuation

This section provides a remarkably simple valuation framework for $\mathfrak{Fat}$, $\mathfrak{Fat}$-α, and $\mathfrak{Fat}$-β tokens. We take inspiration from quantitative finance, but adapt them to use basic techniques that is suitable for the underlying Metheus economy.

$\mathfrak{Fat}$ Valuation

Recall the creditor purchases $\mathfrak{Fat}$ for $1 - \kappa$, with the remaining κ contributed from Metheus operating margins. Rationally, *if* the dividend value of $\mathfrak{Fat}$ is zero, then no buyer would pay more than 1 fiat it light of its fixed collateral value. So 1 fiat is the lower bound on the price ceiling. Now we argue the price floor is 1 fiat using the adage that "*the value of a thing is its arbitrage revalue.*" Under this conceptual trading path, creditor can profit on the Metheus side chain immediately by i.e. selling $\mathfrak{Fat}$ for $1 - \frac{\kappa}{2}$, and pocketing $\frac{\kappa}{2}$ in profit. Such arbitrage trades would continue until all profits disappear. Thus $\mathfrak{Fat}$ is worth *at least* 1 fiat collateral of choice on the side chain. Since trading fee is low on the Metheus side chain, a rational trader who purchased $\mathfrak{Fat}$ with the intention of selling it should sell $\mathfrak{Fat}$ on the side chain to pocket the upside. This sidechain arbitrage action maintain the $\mathfrak{Fat}$ price floor.

$\mathfrak{Fat}$-β Valuation

On the Ethereum mainnet, the margin of κ is too low on account of mainnet transaction costs. However if the creditor hold $\mathfrak{Fat}$-β en bulk, then he can pursue arbitrage opportunity

as above, until the the revalued price of $\mathfrak{Fat}$-$\boldsymbol{\beta}$ is close to 1 fiat. Most importantly, $\mathfrak{Fat}$-$\boldsymbol{\beta}$ can be redeemed for exactly 1 fiat, and can never de-peg modulo TradFi sanction risks.

$\mathfrak{Fat}$-α Valuation

This is the token of interest for an alternative valuation method. Examining the green curve in fig 4.5, it is clear that $\mathfrak{Fat}$-$\boldsymbol{\alpha}$ is in essence a call on the miat-inflation pool. The volume of this pool is a wholistic measure of "temperature" in the Metheus economy. So it appears reasonable to use Black-Scholes to price $\mathfrak{Fat}$-$\boldsymbol{\alpha}$. However we will pursue a different path, this is sensible for a few reasons:

1. (*Audience background*): Black-Scholes was developed by people from the physical sciences. They work in continuous space-time, and express stock-price movement with continuous random walk (Brownian motion) [Shr10]. This monograph's target readers are programmers. They work in discrete time, are more comfortable with tossing coins, traversing upside-down trees, and use simple sums and vanilla integrals.

2. (*System Dynamics*): Black-Scholes places very little assumption on the underlying system model. Metheus's underlying economy is specified exactly in this monograph, so we can and should state the value of $\mathfrak{Fat}$-$\boldsymbol{\alpha}$ as a function of these parameters.

3. (*Practical Considerations*): in reality Black-Scholes is not used to price options anyways. Instead the market prices options, while Black-Scholes is used to compute the implied volatility with the price as an input. [1]

In valuing $\mathfrak{Fat}$-$\boldsymbol{\alpha}$, there are a few assumptions about how the Metheus economy operates:

1. Users cannot burn miat on an ad-hoc basis. This is can be set in the miat-contract, so that miat cannot be sent to the burn address $0x00$.. except by the Metheus admin.

2. Metheus burns miat in an ad-hoc manner, which is not modeled in this section. In practice, burn events will be signaled well ahead of time, so that $\mathfrak{Fat}$-$\boldsymbol{\alpha}$ buyers are aware of its influence on miat-inflation dividends.

3. The dividend payment period of eqn 4.1 is:

   a) roughly in sync with audit period T of QAudit.

   b) Greater than the rate at which Witnesses are rejected in FeaAudit.

   c) Within this duration T, the Auditors and Agents/Witnesses can only liquidate their position at an exchange rate $\mathsf{e}^{\mathsf{t}^{\mathsf{f}}}_{\mathsf{m:f}}$ that is no lesser than the exchange rate $\mathsf{e}^{\mathsf{t}^{o}}_{\mathsf{m:f}}$ they bought into the market. That is:

   $$\mathsf{e}^{\mathsf{t}^{o}}_{\mathsf{m:f}} \leq \mathsf{e}^{\mathsf{t}^{\mathsf{f}}}_{\mathsf{m:f}}, \quad \text{where } \mathsf{t}^{o} < \mathsf{t}^{\mathsf{f}}. \tag{4.2}$$

   This market constraint prevents the unfortunate scenario where the Auditors buys in at 10:1 miat:$\mathfrak{Fat}$ exchange rate, and liquidate at 8:1 exchange rate, and discover that there is not enough money in the fiat-denominated wager pool.

   For example, if $\mathfrak{Fat}$-$\boldsymbol{\alpha}$ expires in 26 weeks, and each dividend payment period is 7 days, then the $\mathfrak{Fat}$-$\boldsymbol{\alpha}$-creditor is paid 26 times from the miat-inflation pool (see fig 4.8). Recall in each payment period, the entire inflation pool is distributed pro-rata to every $\mathfrak{Fat}$-$\boldsymbol{\alpha}$ holder. Given this constraint, we can set T to be 7 days, so that at the end of every week the Agent-submitted models are tested on the hidden test set, and Auditors' fiat wagers are distributed according to miat inflation. This fast turnover is ideal:

[1] That is except in illustrious shops such as Long-Term Capital Management, which despite its name blew up in short order.

a) faster cash turnaround allow Auditors to liquidate their positions, pocket their winnings, and possibly buy back into the market. This increases emotional engagement and increases retention.

b) Faster rate of payout allow Metheus to flood the miat-inflation pool with fiat, so that $\mathfrak{Fat}$-$\boldsymbol{\alpha}$ holders are paid out each week. In the unthinkable arrangement where i.e. T in QAudit is 52 weeks, then $\mathfrak{Fat}$-$\boldsymbol{\alpha}$ holders may never see a cash return on their holdings.

c) Regular payouts significantly simplify the valuation model we present below.

Naturally, QAudit markets do not last 7 days only, recall in 2.7 both the PA and Auditor have a chance to raise the bounty value and reset the QAudit instance.

All in all, these assumptions will be reified as constraints into the Metheus ecosystem calendar. Now we will proceed with a step by step valuation instruction, building towards eqn 4.15. The $\mathfrak{Fat}$-$\boldsymbol{\alpha}$ valuation method is a brute-force sampling regime. It samples all "value-generating-paths" the Metheus market can take over each dividend period, and compute their expected value. This program serves as both:

1. a valuation model for $\mathfrak{Fat}$-$\boldsymbol{\alpha}$;
2. a system dynamics model for the entire Metheus economy.

The system model is framed by four hyper-parameters: $\mathfrak{p}$, $\mathfrak{q}$, μ, and σ, along with proper support over said distributions. The meanings of these parameters are expounded on at length in the main body. The hierarchical sampling process used to compute $\mathfrak{Fat}$-$\boldsymbol{\alpha}$ intrinsic value is summarized in fig 4.9. Now we name the elementary variables and/or entities that underlie the Metheus economy:

1. (time step τ): the *discrete* time step τ is an elementary input into the $\mathfrak{Fat}$-$\boldsymbol{\alpha}$ valuation model. Its numerical value should be selected based on the frequency of audit transactions in the Metheus ecosystem. For example, if every 10 minutes an Auditor or Witness approaches QAudit and FeaAudit respectively, then $\tau = 10$ minutes. Thus τ is a measure of system frequency or hertz.

2. (dividend period $[\tau_\mathsf{o}, \tau_f]$): indexed by time step τ. Given $\tau = 10$ minutes, and every dividend period is 7 days, then $\tau_f = 7 \cdot 24 \cdot 60/10 = 1008$. And we have a sequence of time step indices for *every* dividend period:

$$[\tau_\mathsf{o}, \tau_\mathsf{f}] = \big[\tau_\mathsf{o}, \tau_1, \tau_2, \ldots, \tau_{1007}\big], \quad \text{where } \tau_\mathsf{f} = \tau_{1007}. \tag{4.3}$$

Following the example above, if $\mathfrak{Fat}$-$\boldsymbol{\alpha}$ expires in 6 months, then there are 26 such $[\tau_\mathsf{o}, \tau_\mathsf{f}]$ intervals. The valuation model will estimate the dividend accrued in every step τ of every dividend interval.

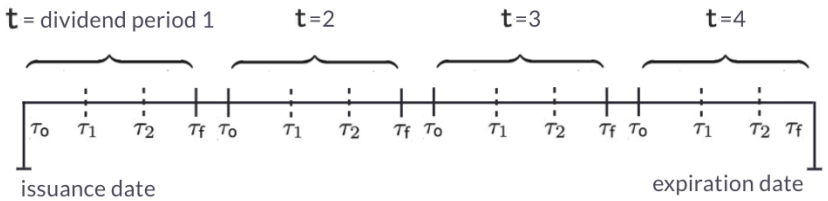


Figure 4.8: In this simple example, $\mathfrak{Fat}$-$\boldsymbol{\alpha}$ expires after 4 dividend periods. Each period has 4 time steps: $[\tau_\mathsf{o}, \tau_1, \tau_2, \tau_\mathsf{f}]$. This valuation method estimates the miat and fiat emission from each time step τ, and compute their expected value.

3. (*exchange rate* $\mathsf{e}^{\tau}_{\mathfrak{m}:\mathfrak{f}}$): the $\mathfrak{miat}$:$\mathfrak{Fat}$ exchange rate as a function of time-step τ. As $\mathfrak{miat}$ is mined, the exchange rate $\mathsf{e}^{\tau}_{\mathfrak{m}:\mathfrak{f}}$ increases. Similarly, as $\mathfrak{Fat}$ is issued, $\mathsf{e}^{\tau}_{\mathfrak{m}:\mathfrak{f}}$ decreases; as $\mathfrak{Fat}$ expires, $\mathsf{e}^{\tau}_{\mathfrak{m}:\mathfrak{f}}$ increases. Thus $\mathsf{e}^{\tau}_{\mathfrak{m}:\mathfrak{f}}$ is not an independent r.v. in the system, but rather a consequence of change in supply of $\mathfrak{miat}$ and $\mathfrak{Fat}$.

4. (AuditUnit): this is an abstraction of both FeaAudit and QAudit market protocols of the last two chapters. That is:

$$\mathsf{AuditUnit} \triangleq \mathsf{QAudit} \mid \mathsf{FeaAudit}.$$

In the context of $\mathfrak{Fat}$-$\boldsymbol{\alpha}$, each AuditUnit instance is in essence a vending machine. At each time step τ, Auditor put in fiat, and receive some amount of fiat back. Concurrently, this AuditUnit vending machine has $\mathfrak{miat}$ flow characterized by figure 4.10.

Figure 4.10: The AuditUnit abstraction and its interaction with the $\mathfrak{miat}$ and fiat collateral pool. 1) On top, the AuditUnit accepts fiat from Auditors, a portion of which is cycled into the fiat collateral pool due to mined $\mathfrak{miat}$. This positive fiat outflow increases overall $\mathfrak{miat}$ supply, and increases the fiat-denominated $\mathfrak{miat}$-inflation pool. On the bottom, a similar story with fiat, except net $\mathfrak{miat}$-flow is zero, so overall $\mathfrak{miat}$ supply does not change. Thus no fiat is placed into the inflation dividend pool.

Every AuditUnit is associated with three random variables (r.v.): 1) $\boldsymbol{b}^{\tau}_{\mathfrak{miat}}$, 2) $\boldsymbol{\omega}^{\tau}_{\mathfrak{miat}}$, and 3) $\mathsf{w}^{\tau}_{\mathsf{fiat}}$. They are sampled in a hierarchical process in the order with which they are listed. [2] At every time step τ, the sampling process proceeds as follows:

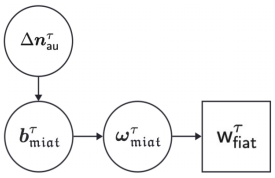


Figure 4.11: On the time interval $[\tau_{\mathsf{o}}, \tau_{\mathsf{f}}]$, a three step sampling process proceeds as follows: 1) first the multinomial r.v. $\Delta\boldsymbol{n}^{\tau}_{\mathsf{au}}$ denoting the fluctuations in the number of AuditUnit is sampled. 2) Then the Bernoulli r.v. $\boldsymbol{b}^{\tau}_{\mathfrak{miat}}$ is sampled, stating whether the Auditor is purchasing newly mined $\mathfrak{miat}$. 3) Next $\boldsymbol{\omega}^{\tau}_{\mathfrak{miat}}$ is sampled from the truncated normal distribution for the Auditor's $\mathfrak{miat}$-denominated order volume. Finally the sampled $\boldsymbol{\omega}^{\tau}_{\mathfrak{miat}}$ value converts to $\mathsf{w}^{\tau}_{\mathsf{fiat}}$ at the current exchange rate $\mathsf{e}^{\tau}_{\mathfrak{m}:\mathfrak{f}}$.

a) (*Bernoulli r.v.* $\boldsymbol{b}^{\tau}_{\mathfrak{miat}} \overset{\mathfrak{q}}{\sim} [0,1]$) modeling the likelihood that the Auditor who approached this AuditUnit does not have sufficient funds in his wallet, so that he buys $\mathfrak{miat}$ to wager. Define:

$$\mathfrak{p}(\boldsymbol{b}^{\tau}_{\mathfrak{miat}} = 1) = \mathfrak{q}. \tag{4.4}$$

If we sample $\boldsymbol{b}^{\tau}_{\mathfrak{miat}} = 0$ then the program stops here. However if $\boldsymbol{b}^{\tau}_{\mathfrak{miat}} = 1$, then the Auditor does not have sufficient $\mathfrak{miat}$. Metheus mines $\mathfrak{miat}$ and sell it to this Auditor. Consequently, this valuation program proceeds to the next step.

b) ($\boldsymbol{\omega}^{\tau}_{\mathfrak{miat}}$ *sampled from the truncated normal distribution*) with support on $\mathbb{R}^{+} \cup \{0\}$:

$$\boldsymbol{\omega}^{\tau}_{\mathfrak{miat}} \sim \mathcal{N}(\mu, \sigma, [0, \mathsf{L})), \tag{4.5}$$

where L is some suitably large number, μ and σ are analyst-chosen values based on historical wagering patterns. This r.v. defines how much $\mathfrak{miat}$ the Auditor

[2] This sampling process resembles latent Dirichlet allocation process for topic modeling in classical natural language processing models.

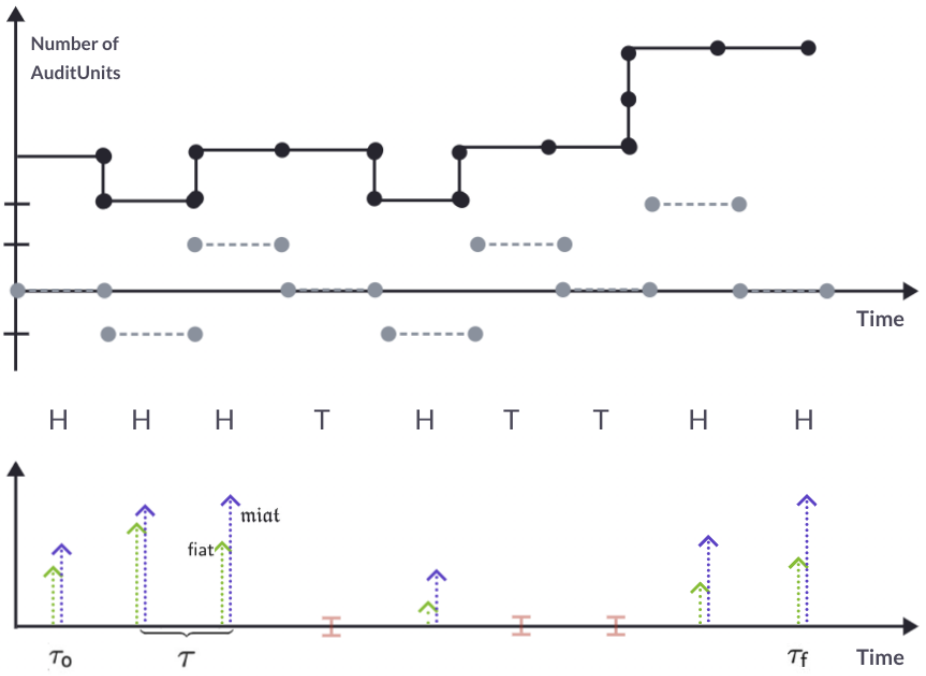


Figure 4.9: $\mathfrak{Fat}$-$\boldsymbol{\alpha}$ is valued by computing its *expected* fiat-denominated dividend yield over all dividend periods. The fixed duration of each period is denoted with $[\tau_0, \tau_f]$. In every $\tau \in [\tau_0, \tau_f]$, an Auditor approaches the AuditUnit and wagers some amount of miat in the market at the current miat:$\mathfrak{Fat}$ exchange rate. This diagram shows the sampling process for *one* dividend paying period.

a) Top: the valuation program first generates an example path of AuditUnits. Denote the number of AuditUnit at time τ with $\boldsymbol{n}^{\tau}_{\mathsf{au}}$, then the delta in this value is modeled with multinomial r.v. $\Delta\boldsymbol{n}^{\tau}_{\mathsf{au}}$. The program samples $\Delta\boldsymbol{n}_{\mathsf{au}}$ $\tau_f - 1$ times (gray dotted lines) to construct the number of AuditUnits over time interval $[\tau_0, \tau_f]$ (discrete black trajectory). For example if $\boldsymbol{n}^{\tau}_{\mathsf{au}} = 10$, then there are 10 AuditUnits operating at time step τ.

b) Middle: τ_f coins with bias q are tossed, expressing the likelihood that the Auditor who approached the AuditUnit has sufficient miat on hand. Denote this Bernoulli r.v. with $\boldsymbol{b}^{\tau}_{\mathfrak{miat}}$, so that $\boldsymbol{b}^{\tau}_{\mathfrak{miat}} = \mathsf{H}$ implies that the Auditor has to buy miat with fiat, thus increasing the total miat supply and contributing to the inflation pool. If $\boldsymbol{b}^{\tau}_{\mathfrak{miat}} = \mathsf{T}$, then no miat is mined and no fiat spent.

c) Bottom: At every step τ where $\boldsymbol{b}^{\tau}_{\mathfrak{miat}} = \mathsf{H}$, sample the Auditor's miat-denominated buy order from the truncated normal distribution with $\boldsymbol{\omega}^{\tau}_{\mathfrak{miat}} \sim \mathcal{N}(\mu, \sigma, [0, \mathsf{L}))$. Finish by computing the fiat amount using the current exchange rate $\mathsf{e}^{\tau}_{\mathfrak{m}:\mathsf{f}}$ updated at each time step τ.

The process above generates the expected miat mined and fiat spent *per* future path within one dividend period. The program then sums over all future paths to compute the expected value of $\mathfrak{Fat}$-$\boldsymbol{\alpha}$ for one dividend paying period. Repeat for N dividend periods and we have $\mathbf{E}\big[\mathfrak{Fat}\text{-}\boldsymbol{\alpha}\big]$, the intrinsic value of $\mathfrak{Fat}$-$\boldsymbol{\alpha}$.

will wager, and $\{0\}$ in the domain expresses an Auditor who declines to wager. Additionally, the initial $\mathfrak{miat}$ value is a constant denoted with $\boldsymbol{\omega}^{\mathsf{o}}_{\mathfrak{miat}}$, it is the global supply of $\mathfrak{miat}$ on the Metheus sidechain at the beginning of dividend period τ_{o}. If we sample $\boldsymbol{\omega}^{\tau}_{\mathfrak{miat}} > 0$, then we can proceed to the final step.

c) (*Induced* fiat *value* $\mathsf{w}^{\tau}_{\mathsf{fiat}}$): the generated value $\boldsymbol{\omega}^{\tau}_{\mathfrak{miat}}$ converts to $\mathsf{w}^{\tau}_{\mathsf{fiat}}$ value at the current exchange rate with:

$$\mathsf{w}^{\tau}_{\mathsf{fiat}} = \frac{\boldsymbol{\omega}^{\tau}_{\mathfrak{miat}}}{\mathsf{e}^{\tau}_{\mathfrak{m}:\mathfrak{f}}}, \quad \text{where } \mathsf{e}^{\tau}_{\mathfrak{m}:\mathfrak{f}} = \frac{\mathrm{sply}(\mathfrak{miat}, \tau)}{\mathrm{sply}(\mathfrak{Fat}\text{-}\boldsymbol{\beta}, \tau)}. \tag{4.6}$$

$\mathsf{w}^{\tau}_{\mathsf{fiat}}$ is the Auditor's fiat-denominated spend at time τ on one AuditUnit. Note since $\mathsf{e}^{\tau}_{\mathfrak{m}:\mathfrak{f}} > 1$ by construction (see fig 4.7), $\mathsf{w}^{\tau}_{\mathsf{fiat}} < \boldsymbol{\omega}^{\tau}_{\mathfrak{miat}}$ for every τ.

d) (Number of audit units $\boldsymbol{n}^{\tau}_{\mathsf{au}}$): the Metheus economy is populated by AuditUnits, they are the elementary settings of production. When PAs initiate new QAudit or FeaAudit markets, $\boldsymbol{n}_{\mathsf{au}}$ increases. When markets wind down and proceeds are distributed, $\boldsymbol{n}_{\mathsf{au}}$ decreases. We model the change in number of AuditUnit with the multinomial r.v. $\Delta\boldsymbol{n}^{\tau}_{\mathsf{au}}$. It takes on discrete value that the analyst may select depending on his expectation of Metheus market conditions. So the range of $\boldsymbol{n}_{\mathsf{au}}$ is a "model hyper-parameter." For example, $\Delta\boldsymbol{n}^{\tau}_{\mathsf{au}}$ may range over $[-1, 0, 1]$, stating that at any time step τ, the number of AuditUnits may decrease by one, remain constant, or increase by 1. Then the analyst may assign discrete probability for every $\Delta\boldsymbol{n}^{\tau}_{\mathsf{au}} \in [-1, 0, 1]$ with:

$$\mathfrak{p}(\Delta\boldsymbol{n}^{\tau}_{\mathsf{au}} = -1) = \mathsf{p}_{-1} \qquad \mathfrak{p}(\Delta\boldsymbol{n}^{\tau}_{\mathsf{au}} = 0) = \mathsf{p}_{0} \qquad \mathfrak{p}(\Delta\boldsymbol{n}^{\tau}_{\mathsf{au}} = 1) = \mathsf{p}_{1}. \tag{4.7}$$

This is effectively a three-sided dice tossed i.i.d. In the simplest case, the distribution may be uniform with $\mathfrak{p}(\Delta\boldsymbol{n}^{\tau}_{\mathsf{au}}) = \frac{1}{3}$ w.r.t the example above. The initial number of AuditUnits in every dividend period is denoted $\boldsymbol{n}^{\mathsf{o}}_{\mathsf{au}}$.

Now we sample one value generating trajectory for the first dividend period:

1. given initial value $\boldsymbol{n}^{\mathsf{o}}_{\mathsf{au}}$ at the moment of $\mathfrak{Fat}\text{-}\boldsymbol{\alpha}$ issuance, generate a $\boldsymbol{n}_{\mathsf{au}}$ trajectory for the first dividend period. This is done by tossing the $\Delta\boldsymbol{n}_{\mathsf{au}}$ dice $\tau_{\mathsf{f}} - 1$ times for this dividend period, generating the time indexed integer sequence $\big[\boldsymbol{n}_{\mathsf{au}}\big]$:

$$\big[\boldsymbol{n}_{\mathsf{au}}\big] = \Big[\boldsymbol{n}^{\tau_{\mathsf{o}}}_{\mathsf{au}}, \boldsymbol{n}^{\tau_{\mathsf{o}}+1}_{\mathsf{au}}, \boldsymbol{n}^{\tau_{\mathsf{o}}+2}_{\mathsf{au}}, \ldots, \boldsymbol{n}^{\tau_{\mathsf{f}}-1}_{\mathsf{au}}, \boldsymbol{n}^{\tau_{\mathsf{f}}}_{\mathsf{au}}\Big],$$
$$\text{where } \boldsymbol{n}^{\tau+1}_{\mathsf{au}} = \boldsymbol{n}^{\tau}_{\mathsf{au}} + \Delta\boldsymbol{n}^{\tau}_{\mathsf{au}}. \tag{4.8}$$

By the i.i.d assumption, the likelihood of this specific trajectory is:

$$\mathfrak{p}\Big(\big[\boldsymbol{n}_{\mathsf{au}}\big]\Big) = \prod_{\tau=\tau_{\mathsf{o}}}^{\tau_{\mathsf{f}}-1} \mathfrak{p}(\Delta\boldsymbol{n}^{\tau}_{\mathsf{au}}). \tag{4.9}$$

Observe that by modeling $\Delta\boldsymbol{n}^{\tau}_{\mathsf{au}}$ instead of $\boldsymbol{n}^{\tau}_{\mathsf{au}}$, we simplify the random experiment. See fig 4.9 or figure below for visuals of this trajectory.

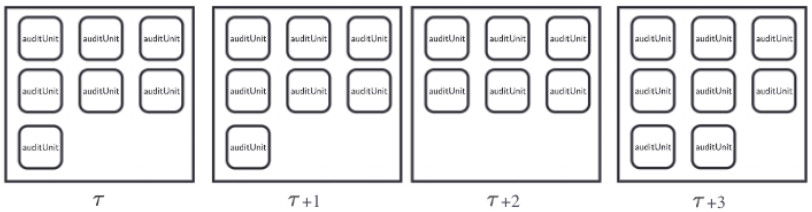


Figure 4.12: The fundamental value drivers in the Metheus are AuditUnits. This valuation method generates possible futures in (de)growth of AuditUnits over dividend period $[\tau_{\mathsf{o}}, \tau_{\mathsf{f}}]$, and estimate the $\mathfrak{miat}$/fiat expenditure in each unit.

2. Next for every AuditUnit in the trajectory $\left[\boldsymbol{n}_{\mathsf{au}}\right]$, toss the q-biased coin τ_{f} times to generate a bit string of length at most τ_{f}:

$$\left[\boldsymbol{b}_{\mathsf{miat}}\right] = \left[\boldsymbol{b}^{\tau_{\mathsf{o}}}_{\mathsf{miat}}, \ldots, \boldsymbol{b}^{\tau_{\mathsf{f}}}_{\mathsf{miat}}\right], \text{ where } \boldsymbol{b}^{\tau}_{\mathsf{miat}} \overset{\mathsf{q}}{\sim} [0,1]. \tag{4.10}$$

In fig 4.9, one such τ_{f}-bit string is shown as HHHT... Observe that if an AuditUnit was added *within* the $[\tau_{\mathsf{o}}, \tau_{\mathsf{f}}]$ interval, then the length of its bit string $\left[\boldsymbol{b}_{\mathsf{miat}}\right]$ is lesser than τ_{f}. We solve this problem by padding it with 0's so that $|\left[\boldsymbol{b}_{\mathsf{miat}}\right]| = \tau_{\mathsf{f}}$. The interpretation is that there is a phantom AuditUnit that the Auditors cannot approach, until it "comes online." The probability of any *singular* τ_{f}-bit string with $0 \leq \mathsf{h} \leq \tau_{\mathsf{f}}$ heads is then:

$$\mathfrak{p}\left(\left[\boldsymbol{b}_{\mathsf{miat}}\right]\right) = \mathsf{q}^{\mathsf{h}}(1-\mathsf{q})^{\tau_{\mathsf{f}}-\mathsf{h}}. \tag{4.11}$$

3. Next for every $\tau \in \left[\tau_{\mathsf{o}}, \tau_{\mathsf{f}}\right]$ and each AuditUnit, compute the expected value of $\boldsymbol{\omega}^{\tau}_{\mathsf{miat}}$ according to eqn 4.5. That is for every $\mathsf{AuditUnit}_{\mathsf{k}}$:

$$\left[\mathbf{E}_{\mathsf{q},\mu,\sigma}\left[\boldsymbol{\omega}_{\mathsf{miat}}\right]\right]_{\mathsf{k}} := \left[\mathbf{E}_{\mathsf{q},\mu,\sigma}\left[\boldsymbol{\omega}^{\tau_{\mathsf{o}}}_{\mathsf{miat}}\right], \mathbf{E}_{\mathsf{q},\mu,\sigma}\left[\boldsymbol{\omega}^{\tau_{\mathsf{o}}+1}_{\mathsf{miat}}\right], \ldots, \mathbf{E}_{\mathsf{q},\mu,\sigma}\left[\boldsymbol{\omega}^{\tau_{\mathsf{f}}-1}_{\mathsf{miat}}\right], \mathbf{E}_{\mathsf{q},\mu,\sigma}\left[\boldsymbol{\omega}^{\tau_{\mathsf{f}}}_{\mathsf{miat}}\right]\right],$$

$$\text{where } \mathbf{E}_{\mathsf{q},\mu,\sigma}\left[\boldsymbol{\omega}^{\tau}_{\mathsf{miat}}\right] = \boldsymbol{b}^{\tau}_{\mathsf{miat}} \int_{\boldsymbol{\omega}_{\mathsf{miat}} \in [0,\mathsf{L}]} \boldsymbol{\omega}_{\mathsf{miat}} \mathbf{f}\left(\boldsymbol{\omega}_{\mathsf{miat}}; \mu, \sigma, [0,\mathsf{L})\right) d\boldsymbol{\omega}_{\mathsf{miat}}. \tag{4.12}$$

Here $\mathbf{f}(\boldsymbol{\omega}_{\mathsf{miat}}; \mu, \sigma, [0,\mathsf{L}))$ is the truncated normal distribution. Each $\mathbf{E}_{\mathsf{q},\mu,\sigma}\left[\boldsymbol{\omega}^{\tau}_{\mathsf{miat}}\right]$ corresponds to a blue arrows in fig 4.9. Repeating this sampling process for every $\mathsf{AuditUnit}_{\mathsf{k}}$ yields set of sequences:

$$\left\{\left[\mathbf{E}_{\mathsf{q},\mu,\sigma}\left[\boldsymbol{\omega}_{\mathsf{miat}}\right]\right]_1, .., \left[\mathbf{E}_{\mathsf{q},\mu,\sigma}\left[\boldsymbol{\omega}_{\mathsf{miat}}\right]\right]_{\mathsf{k}}, .., \left[\mathbf{E}_{\mathsf{q},\mu,\sigma}\left[\boldsymbol{\omega}_{\mathsf{miat}}\right]\right]_{\boldsymbol{n}^{\tau}_{\mathsf{au}}}\right\},$$

This is the miat mint events on every AuditUnits over dividend interval $[\tau_{\mathsf{o}}, \tau_{\mathsf{f}}]$.

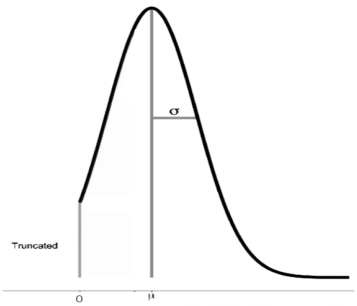


Figure 4.13: For every $\tau \in [\tau_{\mathsf{o}}, \tau_{\mathsf{f}}]$ at each AuditUnit, the expected value of miat-denominated order $\mathbf{E}_{\mathsf{q},\mu,\sigma}\left[\boldsymbol{\omega}^{\tau}_{\mathsf{miat}}\right]$, is calculated by integrating the truncated normal distribution with mean μ, and variance σ^2. In this model, every AuditUnit at each time τ has the same parameters. In a refinement, these values may be a function of each AuditUnit and time step.

4. Given this sequence of miat mint events, we can express the induced sequence of fiat spent across every AuditUnit over the dividend period $[\tau_{\mathsf{o}}, \tau_{\mathsf{f}}]$:

$$\left\{\left[\mathbf{E}_{\mathsf{q},\mu,\sigma}\left[\mathsf{w}_{\mathsf{fiat}}\right]\right]_1, .., \left[\mathbf{E}_{\mathsf{q},\mu,\sigma}\left[\mathsf{w}_{\mathsf{fiat}}\right]\right]_{\mathsf{k}}, .., \left[\mathbf{E}_{\mathsf{q},\mu,\sigma}\left[\mathsf{w}_{\mathsf{fiat}}\right]\right]_{\boldsymbol{n}^{\tau}_{\mathsf{au}}}\right\}.$$

$$\text{where } \left[\mathbf{E}_{\mathsf{q},\mu,\sigma}\left[\mathsf{w}_{\mathsf{fiat}}\right]\right]_{\mathsf{k}} := \left[\mathbf{E}_{\mathsf{q},\mu,\sigma}\left[\mathsf{w}^{\tau_{\mathsf{o}}}_{\mathsf{fiat}}\right], \ldots, \mathbf{E}_{\mathsf{q},\mu,\sigma}\left[\mathsf{w}^{\tau_{\mathsf{f}}}_{\mathsf{fiat}}\right]\right],$$

$$\text{and } \mathbf{E}_{\mathsf{q},\mu,\sigma}\left[\mathsf{w}^{\tau}_{\mathsf{fiat}}\right] = \frac{\mathbf{E}_{\mathsf{q},\mu,\sigma}\left[\boldsymbol{\omega}^{\tau}_{\mathsf{miat}}\right]}{\mathsf{e}^{\tau}_{\mathsf{m:f}}}. \tag{4.13}$$

Note we convert from miat to fiat using instanenous exchange rate τ with:

$$\mathsf{e}^{\tau}_{\mathsf{m:f}} = \frac{\boldsymbol{\omega}^{\mathsf{o}}_{\mathsf{miat}} + \sum_{\mathsf{k}}^{\boldsymbol{n}^{\tau}_{\mathsf{au}}} \mathbf{E}_{\mathsf{q},\mu,\sigma}\left[\boldsymbol{\omega}^{\tau_{\mathsf{o}}}_{\mathsf{miat}}\right] + \ldots + \mathbf{E}_{\mathsf{q},\mu,\sigma}\left[\boldsymbol{\omega}^{\tau}_{\mathsf{miat}}\right]}{\mathrm{sply}(\mathfrak{Fat}\text{-}\boldsymbol{\beta} + 1, \ \tau)}. \tag{4.14}$$

This is the sum over all the $\mathfrak{miat}$ that has been mined across all AuditUnits thus far at time τ, divided by the initial supply of $\mathfrak{Fat}$-$\boldsymbol{\beta}$ plus one. This $\mathsf{e}^{\tau}_{\mathsf{m}:\mathfrak{f}}$ value should then be bounded by the exchange rate floor and celing $(\underline{\mathsf{e}}, \bar{\mathsf{e}})$.

5. Then compute the amount of fiat allocated to the inflation pool at the end of the 1st dividend period τ_{f} with:

$$\bar{\mathfrak{Q}}^{1} = \mathbf{E}_{\mathsf{q},\mu,\sigma}\Big[\mathrm{sply}(\mathsf{w}^{\tau_{\mathsf{f}}}_{\mathsf{fiat}})\Big] = \Big(1 - \frac{\mathsf{e}^{\tau_{\mathsf{o}}}_{\mathsf{m}:\mathfrak{f}}}{\mathsf{e}^{\tau_{\mathsf{f}}}_{\mathsf{m}:\mathfrak{f}}}\Big) \cdot \sum_{\mathsf{i}}^{\boldsymbol{n}_{\mathsf{au}}} \sum_{\tau=\tau_{\mathsf{o}}}^{\tau_{\mathsf{f}}} \mathbf{E}_{\mathsf{q},\mu,\sigma}\big[\mathsf{w}^{\tau}_{\mathsf{fiat}}\big], \quad \text{with } \mathsf{e}^{\tau_{\mathsf{o}}}_{\mathsf{m}:\mathfrak{f}} \leq \mathsf{e}^{\tau_{\mathsf{f}}}_{\mathsf{m}:\mathfrak{f}}.$$

At the end of the first dividend period τ_{f}, AuditUnits liquidates the Auditors' positions, whence $\mathsf{e}^{\tau_{\mathsf{o}}}_{\mathsf{m}:\mathfrak{f}}/\mathsf{e}^{\tau_{\mathsf{f}}}_{\mathsf{m}:\mathfrak{f}}$ fraction of the fiat-denominated wagering pool is inflated away into the dividend pool. Note without inflation, $\bar{\mathfrak{Q}}^{1} = 0$.

6. Now repeat steps (1) - (5) above over all future trajectories on $[\boldsymbol{n}^{\tau}_{\mathsf{au}}]$, $[\boldsymbol{b}^{\tau}_{\mathfrak{miat}}]$, and $[\boldsymbol{\omega}^{\tau}_{\mathfrak{miat}}]$ sequences over $[\tau_{\mathsf{o}}, \tau_{\mathsf{f}}]$. Observe:

   a) $\Delta\boldsymbol{n}_{\mathsf{au}} \in [1..\mathsf{m}]$, so there are $\mathsf{m}^{\tau_{\mathsf{f}}-1}$ possible sequences of $[\boldsymbol{n}_{\mathsf{au}}]$.

   b) $\boldsymbol{b}_{\mathfrak{miat}} \in [0, 1]$, so there are $2^{\tau_{\mathsf{f}}}$ τ_{f}-bit-string of form $[\boldsymbol{b}_{\mathfrak{miat}}]$ for every sequence $[\boldsymbol{n}_{\mathsf{au}}]$.

   c) The values of $\boldsymbol{\omega}^{\tau}_{\mathfrak{miat}}$ are the same in every trajectory since we are using the expected value $\mathbf{E}_{\mathsf{q},\mu,\sigma}\big[\boldsymbol{\omega}^{\tau}_{\mathfrak{miat}}\big]$.

   Therefore the valuation program considers $\mathsf{m}^{\tau_{\mathsf{f}}-1} \times 2^{\tau_{\mathsf{f}}}$ paths of arriving at cumulative dividend pool denominated in fiat, at time $\mathsf{t} = 1$:

$$\big\{\, \bar{\mathfrak{Q}}^{\mathsf{t}}{}_{1}, \ldots, \bar{\mathfrak{Q}}^{\mathsf{t}}{}_{\mathsf{m}^{\tau_{\mathsf{f}}-1} \times 2^{\tau_{\mathsf{f}}}} \big\}.$$

   Define $(\Delta\boldsymbol{n}_{\mathsf{au}})_{\mathsf{r}}$ as rth value of $[1..\mathsf{m}]$, then the expected value of dividends over all trajectories is written:

$$\mathbf{E}_{\mathfrak{p},\mathsf{q},\mu,\sigma}\big[\,\bar{\mathfrak{Q}}\,\big] = \Big[\sum\Big]_{\mathfrak{p}} \sum_{\mathsf{h}=0}^{\tau_{\mathsf{f}}} \binom{\tau_{\mathsf{f}}}{\mathsf{h}} \mathsf{q}^{\mathsf{h}}(1-\mathsf{q})^{\tau_{\mathsf{f}}-\mathsf{h}}\, \bar{\mathfrak{Q}}^{\mathsf{t}}{}_{\mathsf{i}} \qquad \text{where } \bar{\mathfrak{Q}}^{\mathsf{t}}{}_{\mathsf{i}} \in \big\{\, \bar{\mathfrak{Q}}^{\mathsf{t}}{}_{1}, \ldots, \bar{\mathfrak{Q}}^{\mathsf{t}}{}_{\mathsf{m}^{\tau_{\mathsf{f}}-1} \times 2^{\tau_{\mathsf{f}}}} \big\},$$

$$\text{and } \Big[\sum\Big]_{\mathfrak{p}} := \sum_{(\Delta\boldsymbol{n}_{\mathsf{au}})_1=0}^{\tau_{\mathsf{f}}-1} \cdots \sum_{(\Delta\boldsymbol{n}_{\mathsf{au}})_{\mathsf{m}}=0}^{\tau_{\mathsf{f}}-1} \frac{(\tau_{\mathsf{f}}-1)!}{(\Delta\boldsymbol{n}_{\mathsf{au}})_1! \ldots (\Delta\boldsymbol{n}_{\mathsf{au}})_{\mathsf{m}}!} \mathfrak{p}\Big(\big[\boldsymbol{n}_{\mathsf{au}}\big]\Big). \tag{4.15}$$

7. Now this expected dividend pool is allocated pro-rata to every $\mathfrak{Fat}$-$\boldsymbol{\alpha}$ holder at $\mathsf{t} = 1$:

$$\mathbf{E}_{\mathfrak{p},\mathsf{q},\mu,\sigma}\big[\,\mathfrak{Q}^{\mathsf{t}}\big] = \frac{\mathbf{E}_{\mathfrak{p},\mathsf{q},\mu,\sigma}\big[\,\bar{\mathfrak{Q}}^{\mathsf{t}}\big]}{\mathrm{sply}(\mathfrak{Fat}\text{-}\boldsymbol{\alpha},\ \mathsf{t}) + 1}.$$

This is the expected dividend value due to one additional $\mathfrak{Fat}$-$\boldsymbol{\alpha}$ purchased by the analyst at the end of the 1st dividend period τ_{f}, *if* this $\mathfrak{Fat}$-$\boldsymbol{\alpha}$ is minted from one additional $\mathfrak{Fat}$ token issued.

8. The quantity $\mathbf{E}_{\mathfrak{p},\mathsf{q},\mu,\sigma}\big[\,\mathfrak{Q}^{\mathsf{t}}\big]$ is the expected value of dividend paid out to $\mathfrak{Fat}$-$\boldsymbol{\alpha}$ holder in the 1st period, over all possible futures on interval $[\tau_{\mathsf{o}}, \tau_{\mathsf{f}}]$. Now we compute the next period, updating the initial conditions as the average of the previous sample paths:

$$\Big(\mathsf{w}^{\tau_{\mathsf{o}}}_{\mathsf{fiat}}\Big)^{\mathsf{t}+1} = 0 \qquad \Big(\boldsymbol{n}^{\mathsf{o}}_{\mathsf{au}}\Big)^{\mathsf{t}+1} = \Big[\sum\Big]_{\mathfrak{p}} \boldsymbol{n}^{\tau_{\mathsf{f}}}_{\mathsf{au}}$$

$$\Big(\boldsymbol{\omega}^{\mathsf{o}}_{\mathfrak{miat}}\Big)^{\mathsf{t}+1} = \boldsymbol{\omega}^{\mathsf{o}}_{\mathfrak{miat}} + \Big[\sum\Big]_{\mathfrak{p}} \sum_{\mathsf{h}=0}^{\tau_{\mathsf{f}}} \binom{\tau_{\mathsf{f}}}{\mathsf{h}} \mathsf{q}^{\mathsf{h}}(1-\mathsf{q})^{\tau_{\mathsf{f}}-\mathsf{h}} \bar{\omega}_{\mathfrak{miat}}$$

$$\text{where } \bar{\omega}_{\mathsf{miat}} = \sum_{\mathsf{i}}^{\boldsymbol{n}_{\mathsf{au}}} \sum_{\tau=\tau_{\mathsf{o}}}^{\tau_{\mathsf{f}}} \mathbf{E}_{\mathsf{q},\mu,\sigma}\left[\boldsymbol{\omega}_{\mathsf{miat}}^{\tau}\right]$$

$$\left(\mathsf{e}_{\mathsf{m:f}}^{\tau_{\mathsf{o}}}\right)^{\mathsf{t}+1} = \frac{\left(\boldsymbol{\omega}_{\mathsf{miat}}^{\mathsf{o}}\right)^{\mathsf{t}+1}}{\mathrm{sply}(\mathfrak{Fat}\text{-}\boldsymbol{\beta}+1,\ \ \tau_{\mathsf{o}})}. \tag{4.16}$$

Here $\mathsf{w}_{\mathsf{fiat}}^{\tau_{\mathsf{o}}} = 0$ as by construction, the entire dividend pool is paid out. Repeating steps (1) - (7), the program arrives at $\mathbf{E}_{\mathsf{p},\mathsf{q},\mu,\sigma}\left[\wp^{\mathsf{t}+1}\right]$ in the next period.

9. Iterating steps (1) - (8) over all dividend periods, with appropriately updated initial conditions $\boldsymbol{n}_{\mathsf{au}}^{\tau_{\mathsf{o}}}$, $\boldsymbol{\omega}_{\mathsf{miat}}^{\tau_{\mathsf{o}}}$, and $\mathsf{e}_{\mathsf{m:f}}^{\tau_{\mathsf{o}}}$. We arrive at the expected value of $\mathfrak{Fat}\text{-}\boldsymbol{\alpha}$ with:

$$\mathbf{E}_{\mathsf{p},\mathsf{q},\mu,\sigma}\left[\mathfrak{Fat}\text{-}\boldsymbol{\alpha}\right] = \sum_{\mathsf{t}=1}^{\mathsf{N}} \mathbf{E}_{\mathsf{p},\mathsf{q},\mu,\sigma}\left[\wp^{\mathsf{t}}\right] \tag{4.17}$$

where N is the total number of dividend periods for this particular $\mathfrak{Fat}\text{-}\boldsymbol{\alpha}$ token, and the analyst should discount each $\mathbf{E}_{\mathsf{p},\mathsf{q},\mu,\sigma}\left[\mathsf{dividend}_{\mathsf{t}}\right]$ factor with hurdle rate of choice.

Some final commentaries on this valuation method:

1. as promised, the intrinsic value of $\mathfrak{Fat}\text{-}\boldsymbol{\alpha}$ is a statement of the system dynamics of the Metheus economy. Baked within steps (1) - (9) above are all the constraints of the FeaAudit and QAudit market protocols. In particular:
   a) the regular liquidation of wagering positions;
   b) the frequency with which Auditors approach the market;
   c) the protocol's floating exchange rate.
2. In this valuation model, we do not differentiate between AuditUnits or Auditors' behaviors, this is why μ, σ, and q are the same across all units. In reality, some AuditUnits will see more activity than others. So their respective parameters may differ. However this will also make the valuation model more computationally complex.
3. Because all the values are fixed a-priori, the values of eqn 4.15 and eqns 4.16 do not need to be "sampled." Instead their value can be computed quickly and exactly. However if the analyst model changes in parameters, then the framework requires true sampling.
4. Instead of resetting the initial condition as 4.16 suggests, it is possible to express the entire trajectory of $[\boldsymbol{n}_{\mathsf{au}}]$, $[\boldsymbol{b}_{\mathsf{miat}}]$, $[\boldsymbol{\omega}_{\mathsf{miat}}]$, and $[\mathsf{w}_{\mathsf{fiat}}]$ over all dividend paying periods. So that each trajectory is of length $\mathsf{N} \times (\tau_{\mathsf{f}} - \tau_{\mathsf{o}})$, and i.e. the number of trajectories in $[\boldsymbol{b}_{\mathsf{miat}}]$ is $2^{\mathsf{N}\times(\tau_{\mathsf{f}}-\tau_{\mathsf{o}})}$. Then the valuation program would walk down the N dividend paying periods, and compute the dividend owed in that interval.
5. This model assumes $\Delta\boldsymbol{n}_{\mathsf{au}}$, $\boldsymbol{b}_{\mathsf{miat}}$, and $\boldsymbol{\omega}_{\mathsf{miat}}$ are independent. In practice they do depend on each other in ways that is difficult to express, so the principle of independence minimizes the number of assumptions the model has to make.
6. What is not modeled in this method is the increase in supply of $\mathfrak{Fat}\text{-}\boldsymbol{\beta}$. The analyst may find their own model for increase in supply per market conditions, and enter into the last expression in eqn 4.16 in $\mathrm{sply}(\mathfrak{Fat}\text{-}\boldsymbol{\beta}, \tau_{\mathsf{o}})$.
7. Estimated $\mathfrak{Fat}\text{-}\boldsymbol{\alpha}$ value may differ greatly depending on the choice of:
   a) the hyper-parameter set $\{\mu, \sigma, \mathsf{q}, \mathsf{p}\}$;
   b) the support of p in $\Delta\boldsymbol{n}_{\mathsf{au}} \in [1...\mathsf{m}]$;
   c) the range $[0, \mathsf{L})$ the truncated normal distribution is defined over.

Large range of estimated values is ideal as generous $\mathfrak{Fat}$ spreads stimulate trading activity. Recall $\mathfrak{Fat}$-$\boldsymbol{\alpha}$ is a retail product, it gives the average folk a chance to speculate on the future of the AI economy.

Remark 4.4.1. (*Alternative* $\mathfrak{Fat}$-$\boldsymbol{\alpha}$ *mechanism*). Instead of regular miat-inflation dividend payouts, Metheus can withhold distribution, and become a $\mathfrak{Fat}$-$\boldsymbol{\alpha}$ market-maker of last resort. Thus Metheus is both the $\mathfrak{Fat}$-$\boldsymbol{\alpha}$ issuer and a $\mathfrak{Fat}$-Amm with this arrangement:

4.4.1.1. issue $\mathfrak{Fat}$, $\mathfrak{Fat}$-$\boldsymbol{\beta}$ and $\mathfrak{Fat}$-$\boldsymbol{\alpha}$ as before. In particular, $\mathfrak{Fat}$-$\boldsymbol{\beta}$ still has voting rights, and $\mathfrak{Fat}$-$\boldsymbol{\alpha}$ has an expiration date attached as fig 4.8 shows.

4.4.1.2. Collect fiat-denominated inflation into the inflation pool, but do not distribute.

4.4.1.3. At any point in time τ *before* the $\mathfrak{Fat}$-$\boldsymbol{\alpha}$ expiration date, its holder may approach the $\mathfrak{Fat}$-$\boldsymbol{\alpha}$ Amm and sell it to Metheus at this price:

$$\mathsf{price}(\mathfrak{Fat}\text{-}\boldsymbol{\alpha}) = \frac{\mathrm{sply}(\text{fiat inflation pool}, \tau)}{\mathrm{sply}(\mathfrak{Fat}\text{-}\boldsymbol{\alpha},\ \tau)}.$$

That is to say the Amm price of $\mathfrak{Fat}$-$\boldsymbol{\alpha}$ is the pro-rata distribution rate. The fiat value of $\mathfrak{Fat}$-$\boldsymbol{\alpha}$ in the dividend vs. buy-back scenarios are comparable modulo discount factor. Hence the valuation method developed before still applies.

4.5 Conclusion

This chapter specifies the Fair, Accountable, and Transparent stablecoin, or $\mathfrak{Fat}$ stablecoin. It is fully collateralized by fiat of choice, whereby both Metheus and the $\mathfrak{Fat}$-creditor contribute to the collateral pool. This arrangement allows $\mathfrak{Fat}$-holders to buy 1 unit of fiat at a discounted value, and gives Metheus a persistent source of leverage. Furthermore, $\mathfrak{Fat}$ factors into $\mathfrak{Fat}$-$\boldsymbol{\alpha}$ expressing the miat-inflation dividend, and $\mathfrak{Fat}$-$\boldsymbol{\beta}$ claiming the underlying collateral of exactly 1 fiat. In an ideal scenario, Metheus-Auditors are also $\mathfrak{Fat}$-creditors. This aligns the incentive end-to-end, so that Auditors are committed to the platform's long term success, and receive commensurate $\mathfrak{Fat}$ upside from value-generating activities ecosystem wide.

Additionally $\mathfrak{Fat}$-$\boldsymbol{\beta}$ is a governance token, its holders define the miat:$\mathfrak{Fat}$ exchange rate envelope. The Metheus operating model hinges on fluctuations in this exchange rate, so $\mathfrak{Fat}$-$\boldsymbol{\beta}$ holders directly control how much of the Auditors' collective wagers are returned to the winners, vs. allocation to $\mathfrak{Fat}$ social security. The chapter proceeds to value $\mathfrak{Fat}$-$\boldsymbol{\alpha}$ and $\mathfrak{Fat}$-$\boldsymbol{\beta}$ as a function of the underlying economy. It argues the arbitrage revalue of $\mathfrak{Fat}$-$\boldsymbol{\beta}$ tokens is 1 fiat even though its initial sale price is less than one. Then we express the intrinsic value of $\mathfrak{Fat}$-$\boldsymbol{\alpha}$ as a consequence of Metheus's underlying system dynamics. The $\mathfrak{Fat}$-$\boldsymbol{\alpha}$ valuation framework is the system model.

4.6 Postscript

In so far as we know, $\mathfrak{Fat}$ is the first **entropy stablecoin**. Its rate of issuance and $\mathfrak{Fat}$ upside is a direct function of the underlying machine-intelligence economy. This point is critical, whereas most stablecoins are based on pure speculation, or just passively accept deposits, the $\mathfrak{Fat}$ stablecoin mediates a true economy. "Auditing" then is a wholistic way to measure the myriads of achievements within this sector. $\mathfrak{Fat}$ holders are not only believers of machine-intellgience as a transformational computing paradigm, they are in fact pushing it forward by broadening the horizon of what is possible. Together we invest in more models, more data, more talent, and more mechanisms of governing the future of this technology.

Appendices

APPENDIX A

Monograph Notation

Elementary Notions

Notation	Description
Amm	Automated market maker
creditor	buyer of $\mathfrak{Fat}$ stablecoins
$\mathsf{Agent}_\mathsf{k}$	the kth Agent
$\mathsf{Auditor}_\mathsf{r}$	the rth Auditor
PA	the principle Auditor
Wit_k	the kth Witness
VA	the vice Auditor or eunuch
Cohort	a set of Witnesses marked by demographic data
AuditUnit	either an instance of FeaAudit or QAudit market protocol
τ	one time increment within one stage, i.e. 10 minutes
t	one stage, may be i.e. 24 hours
T	total number of stages, may be i.e. 7 days
fiat	any fiat of choice
$\mathfrak{miat}$	the *Machine Intelligence Audit Toke*n, a ERC20 token on the Metheus sidechain
$\mathfrak{mint}^\mathsf{t}_\mathsf{k}$	the *Machine Intelligence Nonfungible Token* for the kth model at stage t, a ERC1155 nonfungible token on the Metheus side chain
$\mathfrak{feat}^\mathsf{t}_\mathsf{k}$	the *Fairness, Equality, and Equity token* for the kth Witness at stage t a ERC721 nonfungible token on the Metheus side chain
$\mathfrak{Fat}$	the *Fair Accountable Transparent stablecoin*, pegged to 1 fiat of choice
$\mathfrak{Fat}$-$\boldsymbol{\beta}$	the $\mathfrak{Fat}$ ERC20 stablecoin pegged to 1 fiat of choice on the Ethereum mainnet
$\mathfrak{Fat}$-$\boldsymbol{\alpha}$	the $\mathfrak{Fat}$ high-variance ERC721 nonfungible token on the Ethereum mainnet
κ	Metheus's contribution into the fiat-denominated $\mathfrak{Fat}$ collateral pool

Table A.1: Notation used for elementary market entities in Metheus.

FeaAudit Notation

Notation	Description
$\mathfrak{v}^\mathsf{t}_\mathsf{k}$	Wit_k's or VA_k's $\mathfrak{miat}$-denominated wealth at stage t
$\xi \in (0,1)$	wealth seizure parameter

Table A.2: Notation used for all FeaAudit market entities in Metheus.

QAudit Notation

Notation	Description
$\mathfrak{v}_{\mathsf{pa}}$	PA-defined bounty value
$\mathfrak{v}_{\mathsf{k}}^{\mathsf{t}}$	reward given to $\mathsf{Agent}_{\mathsf{k}}$ or $\mathsf{Auditor}_{\mathsf{k}}$ at stage t
$\varrho_{\mathsf{k}}^{\mathsf{t}}$	regret taken from $\mathsf{Auditor}_{\mathsf{k}}$ at stage t
$\mathbf{s}^{\mathsf{t}}$	QAudit Amm's share vector $\left(\mathsf{s}_1^{\mathsf{t}}, \ldots, \mathsf{s}_{\mathsf{K}}^{\mathsf{t}}\right)$ at stage t
$\mathbf{b}_{\mathsf{k}}^{\mathsf{t}}$	$\mathsf{Auditor}_{\mathsf{k}}$'s buy vector $\left(\mathsf{b}_1^{\mathsf{t}}, \ldots, \mathsf{b}_{\mathsf{K}}^{\mathsf{t}}\right)$ at stage t
$\boldsymbol{\rho}$	price vector $\left(\rho_1^{\mathsf{t}}, \ldots, \rho_{\mathsf{K}}^{\mathsf{t}}\right)$ of all models at stage t
$\mathsf{v}_{\rho}^{\mathsf{t}}$	Dutch price ascending price factor at stage t applied to all models
$\curlyvee \in (0,1)$	regime change risk which functions as a discount factor
$\mathbb{D}_{\mathsf{k}}^{\mathsf{t}}$	generic source of data (i.e. model $\boldsymbol{\theta}$ or dataset $\mathbb{D}$) reported by $\mathsf{Agent}_{\mathsf{k}}$ at stage t
$\mathbb{D}_{\mathsf{train}}$	public training set of form $\{(\mathbf{x}, \mathsf{y})_1, \ldots\}$
$\mathbb{D}_{\mathsf{test}}$	hidden test set used for final judgment of models
$\mathbb{D}_{\mathsf{val}}$	public validation set for initial ranking of models
$\boldsymbol{\eta}$	noise function that emits data for discriminative model

Table A.3: Notation used for all QAudit market entities in Metheus.

$\mathfrak{Fat}$ Notation

Notation	Description
$\mathsf{e}_{\mathsf{m:f}}^{\tau}$	$\mathfrak{miat}$:$\mathfrak{Fat}$ exchange rate at time step τ
$\boldsymbol{b}_{\mathfrak{miat}}^{\tau}$	r.v. denoting whether some Auditor is purchasing $\mathfrak{miat}$ with fiat
$\boldsymbol{\omega}_{\mathfrak{miat}}^{\tau}$	r.v denoting the amount of $\mathfrak{miat}$ spent by Auditor
$\boldsymbol{\omega}_{\mathfrak{miat}}^{\mathsf{o}}$	constant denoting the initial supply of $\mathfrak{miat}$ at time τ_{o}
$\mathsf{w}_{\mathsf{fiat}}^{\tau}$	r.v. denoting the amount of fiat spent by Auditor
$\boldsymbol{n}_{\mathsf{au}}^{\tau}$	r.v. denoting the total number of AuditUnit at time τ
$\boldsymbol{n}_{\mathsf{au}}^{\mathsf{o}}$	constant denoting the initial number of AuditUnit at time τ
$\bar{\varrho}^{\mathsf{t}}$	expected value of the global fiat-denominated inflation dividend pool at dividend period t
ϱ^{t}	expected dividend value per share of $\mathfrak{Fat}$ stablecoin at dividend period t

Table A.4: Notation used for all $\mathfrak{Fat}$-related market entities in Metheus.

Statistics Notation

Notation	Description
$\mathsf{y} \in \{0,1\}$	binary label on $\mathbf{x}$
$\mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathbb{V}$	some arbitrarily-valued vector input in some space $\mathbb{V}$
$\phi_{\mathbf{x}} \in \mathbb{R}^{\mathsf{d}_0}$	real-valued embedding of $\mathbf{x}$ mapped onto feature space of dimension d_0 by the function ϕ
$\boldsymbol{\theta}_{\mathsf{k}}^{\mathsf{t}} \in \mathbb{R}^{\mathsf{d} \times \mathsf{n} \times \mathsf{h}}$	a generative model of form $\mathfrak{p}(\phi_{\mathbf{x}_\tau} \vert \phi_{\mathbf{x}_{\tau-1}}; \boldsymbol{\theta}_{\mathsf{k}}^{\mathsf{t}})$ submitted by the kth Agent at stage t, with input $\phi_{\mathbf{x}_{\tau-1}}$, and generated output $\phi_{\mathbf{x}_\tau}$
$\mathbf{w}_{\mathsf{k}}^{\mathsf{t}} \in \mathbb{R}^{\mathsf{d}_0 \times \mathsf{n}}$	parameterizes discriminative model of form $\mathfrak{p}(y \vert \phi_{\mathbf{x}}; \mathbf{w}_{\mathsf{k}}^{\mathsf{t}})$ from modeled trained on the kth $\mathsf{Agent}_{\mathsf{k}}$'s submission at stage t
$\mathsf{p}_{\mathsf{k}}^{\mathsf{t}}, \mathsf{q}_{\mathsf{r}}^{\mathsf{t}} \in [0,1]$	scalar probability values submitted by kth or rth Agent at stage t
$\mathbf{p} \in [0,1]^{\vert\Omega\vert}$	probability vector over set abstract sample space Ω
$\mathfrak{p} : \mathbb{R}^{\mathsf{d}_0} \to [0,1]$	probability function of form $\mathfrak{p}(\phi_{\mathbf{x}})$
$\mathsf{coin}_{\mathsf{p}}$	Bernoulli r.v. with bias p

Table A.5: Notation used for all statistics and machine learning related entities.

APPENDIX B

Definitions

All book definitions are found here.

B.1 Randomized Algorithms

Definition B.1.1. (*Randomized Algorithm*). A randomized algorithm $\mathcal{A}$ with domain $\mathbb{A}$ and range $\mathbb{B}$ is associated with a mapping $M : \mathbb{A} \to \Delta(\mathbb{B})$. On input $\mathbf{a} \in \mathbb{A}$, the algorithm $\mathcal{A}$ outputs $\mathcal{A}(\mathbf{a}) = \mathbf{b}$ with probability $(M(\mathbf{a}))_b$ for every $\mathbf{b} \in \mathbb{B}$. The probability space is over the coin flips of algorithm $\mathcal{A}$ [DR14, p. 16]. Here the *probability simplex* on discrete set $\mathbb{B}$, denoted $\Delta(\mathbb{B})$ is defined with:

$$\Delta(\mathbb{B}) = \left\{ \mathbf{x} \in \mathbb{R}^{|\mathbb{B}|} \ : \ \mathsf{x}_\mathsf{i} \geq 0 \ \text{ for every } \mathsf{i} \ \text{ and } \sum_{\mathsf{i}=1}^{|\mathbb{B}|} \mathsf{x}_\mathsf{i} = 1 \right\}.$$

Definition B.1.2. (*Differential Privacy*). A randomized algorithm $\mathcal{A}$ with domain $\mathbb{N}^{\mathcal{X}}$ is (ϵ, δ)-differentially private if for all $\mathbb{S} \subseteq \mathsf{Range}(\mathcal{A})$ and for all $\mathbf{x}, \mathbf{y} \in \mathbb{N}^{\mathcal{X}}$ s.t. $\|\mathbf{x} - \mathbf{y}\|_1 \leq 1$:

$$\mathfrak{p}\Big(\mathcal{A}(\mathbf{x}) \in \mathbb{S}\Big) \leq \exp(\epsilon)\mathfrak{p}\Big(\mathcal{A}(\mathbf{y}) \in \mathbb{S}\Big) + \delta,$$

where the probability space is over the coin flips of the mechanism $\mathcal{A}$. If $\delta = 0$, then we say $\mathcal{A}$ is ϵ-differentially private [DR14, p. 17].

Definition B.1.3. (*Chernoff Bound for Binomial Distribution*). Let $X \sim \mathsf{Bin}(\mathsf{N}, \mathsf{p})$ and let $\mu = \mathbf{E}[X]$. For any $0 < \delta < 1$:

B.1.3.1. Upper tail bound:

$$\mathfrak{p}\big(X \geq (1+\delta)\mu\big) \leq \mathsf{exp}\left(-\frac{\delta^2\mu}{3}\right).$$

B.1.3.2. Lower tail bound:

$$\mathfrak{p}\big(X \leq (1-\delta)\mu\big) \leq \mathsf{exp}\left(-\frac{\delta^2\mu}{2}\right).$$

Definition B.1.4. (*Random Walk and Markov Chain*) A random walk is a process for traversing a graph where at every step we follow an outgoing edge chosen uniformly at random. A Markov chain is similar except the outgoing edge is chosen according to an arbitrary fixed distribution. An important property of a Markov chain is the memorylessness property: the future behavior of a Markov chain depends only on its current state, and not on how it arrived at the present state. So that given r.v. X indexed by time τ, we have:

$$\mathfrak{p}(X = \mathbf{x}_{\tau+1} | \mathbf{x}_o, \ldots, \mathbf{x}_\tau) = \mathfrak{p}(X = \mathbf{x}_{\tau+1} | \mathbf{x}_\tau).$$

[MR95, p. 129].

Definition B.1.5. (*Hitting time*) The hitting time h_{uv} (sometimes called the mean first passage time) is the expected number of steps in a random walk over graph with vertices u and v that starts at u and ends upon first reaching v [MR95, p. 133].

Definition B.1.6. (*Markov Chain Transition Matrix*) Given r.v. X_τ indexed by time τ denoting the probability that the current state is at $\mathfrak{s}_\mathsf{r}$ at step τ, and a probability vector where:

$$\mathbf{p}_\tau = \big([\mathbf{p}_\tau]_1, \ldots, [\mathbf{p}_\tau]_\mathsf{r}, \ldots [\mathbf{p}_\tau]_\mathsf{K}\big) \qquad \mathfrak{p}(X_\tau = \mathfrak{s}_\mathsf{r}) = [\mathbf{p}_\tau]_\mathsf{r} \qquad \sum_{r=1}^{\mathsf{K}} [\mathbf{p}_\tau]_\mathsf{r} = 1.$$

We can associate it with a state transition matrix $\mathbf{M}$, or the "walk matrix." The Walk matrix has dimension $K \times K$, so that every entry $[\mathbf{M}]_{\mathsf{rs}}$ is the probability of transition from state $\mathfrak{s}_\mathsf{r}$ to state $\mathfrak{s}_\mathsf{s}$:

$$[\mathbf{M}]_{\mathsf{rs}} = \mathfrak{p}(X_{\tau+1} = \mathfrak{s}_\mathsf{s} | X_\tau = \mathfrak{s}_\mathsf{r}).$$

Additionally, the walk matrix $\mathbf{M}$ is row-stochastic in that the sum of entries in every row r must be 1, corresponding to the likelihood of exiting state $\mathfrak{s}_\mathsf{r}$ in the Markov chain. Given this matrix, we can write one step of the random walk with $\mathbf{p}_{\tau+1} = \mathbf{p}_\tau \mathbf{M}$, and n steps of the walk with: $\mathbf{p}_n = \mathbf{p}_o \mathbf{M}^n$.

Definition B.1.7. (*Stationary Distribution*) for the Markov chain with transition matrix $\mathbf{M}$ is written: $\mathbf{p} = \mathbf{p}\mathbf{M}^n$, this is the steady state of the function $\mathbf{M}$ [MR95, p. 132].

Theorem B.1.8. *(Fundamental Theorem of Markov Chains) An irreducible, finite, and ergodic Markov chain has properties:*

B.1.8.1. *all states are ergodic, meaning it is ergodic.*

B.1.8.2. *There is a unique stationary distribution $\boldsymbol{p}$ s.t.*

$$\boldsymbol{p}^\intercal = \boldsymbol{p}\mathbf{M}, \tag{B.1}$$

where $\mathbf{M}$ is the walk matrix, and observe that $\boldsymbol{p}$ is the principle eigenvector of $\mathbf{M}$ with eigenvalue $\lambda_1 = 1$.

B.1.8.3. *The hitting time of any state $\mathfrak{s}_\mathsf{r}$ is $h_{\mathsf{sr}} = \frac{1}{[\boldsymbol{p}^*]_r}$.*

B.1.8.4. *let $N(r, \tau)$ be the number of times the Markov chain visit state r in τ steps, then:*

$$\lim_{\tau \to \infty} \frac{N(r, \tau)}{\tau} = [\boldsymbol{p}^*]_r.$$

Finally, we know that the first eigenvalue of $\mathbf{M}$ λ_1 is always equal to 1, and the second eigenvalue λ_2 determines the mixing time, or the rate at which repeated applications of $\mathbf{M}$ at any $\boldsymbol{p}$ converges to the stationary distribution [MR95, p. 132].

B.2 Game Theory

The follow set of equilibrium definitions are drawn from [Rou14], and are oriented around the notion of *potential games*.

Definition B.2.1. (*Pure Nash Equilibria*) A strategy profile $\mathbf{s}$ of a cost-minimization game is a *pure Nash equilibrium* (PNE) if for every player $\mathsf{i} \in \{1, \ldots, \mathsf{K}\}$ and every uniliteral deviation $\mathsf{s}_\mathsf{i} \in \mathbb{S}_\mathsf{i}$:

$$\mathbf{C}_\mathsf{i}(\mathbf{s}) \leq \mathbf{C}_\mathsf{i}(\mathsf{s}'_\mathsf{i}, \mathbf{s}_{-\mathsf{i}}).$$

Definition B.2.2. (*Mixed Nash Equilibria*) Distributions $\mathfrak{p}_1, \ldots, \mathfrak{p}_K$ over strategy sets $\mathbb{S}_1, \ldots, \mathbb{S}_K$ of a cost-minimization game constitutes a *Mixed Nash Equilibria* (MNE) if for every player $i \in \{1, \ldots, K\}$ and every uniliteral deviation $s_i \in \mathbb{S}_i$:

$$\mathbf{E}_{s\sim\mathfrak{q}}\big[\mathbf{C}_i(\mathbf{s})\big] \leq \mathbf{E}_{s\sim\mathfrak{q}}\big[\mathbf{C}_i(s_i', \mathbf{s}_{-i})\big],$$

where $\mathfrak{q} := \mathfrak{p}_1 \times \ldots \times \mathfrak{p}_K$.

Definition B.2.3. (*Correlated Nash Equilibria*) A distributions $\mathfrak{q}$ on the set $\mathbb{S}_1, \ldots, \mathbb{S}_K$ of outcomes of a cost-minimization game is a *Correlated Nash Equilibria* (CE) if for every player $i \in \{1, \ldots, K\}$ and every uniliteral deviation $s_i \in \mathbb{S}_i$:

$$\mathbf{E}_{s\sim\mathfrak{q}}\big[\mathbf{C}_i(\mathbf{s})|s_i\big] \leq \mathbf{E}_{s\sim\mathfrak{q}}\big[\mathbf{C}_i(s_i', \mathbf{s}_{-i})|s_i\big].$$

Definition B.2.4. (*Coarse Correlated Equilibria*) A distributions $\mathfrak{q}$ on the set $\mathbb{S}_1, \ldots, \mathbb{S}_K$ of outcomes of a cost-minimization game is a *Coarse Correlated Equilibria* (CCE) if for every player $i \in \{1, \ldots, K\}$ and every uniliteral deviation $s_i \in \mathbb{S}_i$:

$$\mathbf{E}_{s\sim\mathfrak{q}}\big[\mathbf{C}_i(\mathbf{s})\big] \leq \mathbf{E}_{s\sim\mathfrak{q}}\big[\mathbf{C}_i(s_i', \mathbf{s}_{-i})\big].$$

Definition B.2.5. (*Price of Anarchy*) of a game w.r.t social function f is:

$$\frac{\min_{s,NE} f(s)}{\mathsf{OPT}} \text{ for utilities} \quad \text{and} \quad \frac{\max_{s,NE} f(s)}{\mathsf{OPT}} \text{ for costs.}$$

NE is Nash Equilibrium. The sum of social welfare is written:

$$f(s) = \sum_i \mathbf{u}_i(s).$$

And the optimal solution is:

$$\mathsf{OPT} = \max_{s\in\mathbb{S}} f(s).$$

Here $s = (s_1, \ldots, s_n)$ denotes the joint vector of strategies selected by n players in the space of $\mathbb{S}_1 \times \ldots \times \mathbb{S}_n$ actions. With game utility/cost function $\mathbf{u}_i : \mathbb{S} \to \mathbb{R}$ for every player i, at any joint action $s \in \mathbb{S}$.

APPENDIX C

Proofs

This section provides all proofs from the main body.

C.1 Agents-Agents Sub-Game

Proof. for lemma 2.5.1. Recall $\mathsf{Agent_r}$'s objective is to report one of the top R models that can participate in rewards, and maximize his profit with:

$$\arg\max_{\mathbf{w}}\big(\mathfrak{v}^{\mathsf{t}}_{\mathsf{r}} - \mathsf{cost}^{\mathsf{t}}_{\mathsf{r}}\big) = \arg\max_{\mathbf{w}}\big(\beta_{\mathsf{r}} \cdot \mathfrak{v}^{\mathsf{t}} - \mathsf{cost}^{\mathsf{t}}_{\mathsf{r}}\big)$$

$$\text{where } \beta_{\mathsf{r}} = \frac{e^{\mathsf{logprob}^{\mathsf{t}}_{\mathsf{r}}}}{e^{\mathsf{logprob}^{\mathsf{t}}_{1}} + \ldots + e^{\mathsf{logprob}^{\mathsf{t}}_{\mathsf{R}}}} \text{ and } e^{\mathsf{logprob}^{\mathsf{t}}_{\mathsf{r}}} = \prod_{(\phi_{\mathbf{x}},\mathsf{y})\in\mathbb{D}} \mathfrak{p}(\mathsf{y}|\phi_{\mathbf{x}};\mathbf{w}^{\mathsf{t}}_{\mathsf{r}}).$$

The value $\mathfrak{v}^{\mathsf{t}}$ is some constant that is wagered on all the losing models in this current stage t, and $\mathsf{cost}^{\mathsf{t}}_{\mathsf{r}}$ is some fixed cost of creating any model $\mathbf{w}$. Setting $\mathsf{q} = \exp(\mathsf{logprob}^{\mathsf{t}}_{\mathsf{r}})$, the expression above can be rewritten as:

$$\arg\max_{\mathbf{w}} \frac{\mathsf{q}}{\mathsf{q}+C} \cdot \mathfrak{v}^{\mathsf{t}} - \mathsf{cost}^{\mathsf{t}}_{\mathsf{r}} \text{ where } C = \sum_{\mathsf{s}\in\{1..\mathsf{R}\}/\{\mathsf{r}\}} \exp(\mathsf{logprob}^{\mathsf{t}}_{\mathsf{s}}) \tag{C.1}$$

It is self evident that $\mathsf{Agent_r}$'s solution is to maximize data-likelihood q, if neither $\mathsf{cost}^{\mathsf{t}}_{\mathsf{r}}$ nor $\mathfrak{v}^{\mathsf{t}}$ depend on the values of $\mathbf{w}^{\mathsf{t}}_{\mathsf{r}}$.

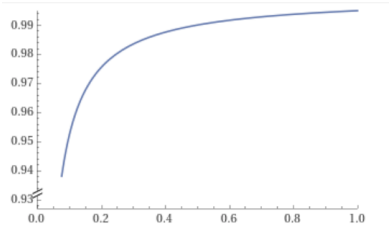


Figure C.1: Expression C.1 plotted with $C = 0.005$, $\mathfrak{v}^{\mathsf{t}} = 1$, and $cost^{\mathsf{t}}_{\mathsf{r}} = 0$. Note the expression is monotonically increasing in $\mathsf{q} = \exp(\mathsf{logprob}^{\mathsf{t}}_{\mathsf{r}})$, stating $\mathsf{Agent_r}$ can maximize his share of winning wagers by maximizing data likelihood.

Moreover, we can expand C.1 modulo the cost factor into a training objective and determine how the Agent should behave with this new objective:

$$\arg\max_{\mathbf{w}} \frac{\mathsf{q}}{C+\mathsf{q}} \cdot \mathfrak{v}^{\mathsf{t}} = \arg\max_{\mathbf{w}} \left(\frac{1}{\mathsf{q}}C + 1\right)^{-1} \cdot \mathfrak{v}^{\mathsf{t}}$$
$$\overset{log}{\Longrightarrow} \arg\max_{\mathbf{w}} \left(\log \mathfrak{v}^{\mathsf{t}} - \log\left(1 + \frac{C}{\prod \mathfrak{p}(\mathsf{y}|\phi_{\mathbf{x}};\mathbf{w}^{\mathsf{t}}_{\mathsf{r}})}\right)\right).$$

Note $\log \mathfrak{v}^{\mathsf{t}}$ is a constant ≥ 0, and as the data likelihood term $\prod \mathfrak{p}(\mathsf{y}|\phi_{\mathbf{x}};\mathbf{w}^{\mathsf{t}}_{\mathsf{r}})$ increases with better $\mathbf{w}$, the overall value of the objective function increases. That is to say $\mathsf{Agent_r}$ should

select $\mathbf{w}_{\mathsf{r}}^{\mathsf{t}}$ over $\mathbf{v}_{\mathsf{r}}^{\mathsf{t}}$ as:

$$\log \mathfrak{v}^{\mathsf{t}} - \log\big(1 + C \cdot \big(\prod \mathfrak{p}(\mathsf{y}|\phi_{\mathbf{x}}; \mathbf{w}_{\mathsf{r}}^{\mathsf{t}})\big)^{-1}\big) > \log \mathfrak{v}^{\mathsf{t}} - \log\big(1 + C \cdot \big(\prod \mathfrak{p}(\mathsf{y}|\phi_{\mathbf{x}}; \mathbf{v}_{\mathsf{r}}^{\mathsf{t}})\big)^{-1}\big),$$

per assumption of expression 2.1. Alternatively, we can find a more familiar expression:

$$\begin{aligned}
&\arg\max_{\mathbf{w}} \frac{\mathsf{q}}{C + \mathsf{q}} \cdot \mathfrak{v}^{\mathsf{t}} \\
&\arg\max_{\mathbf{w}} \prod \mathfrak{p}(\mathsf{y}|\phi_{\mathbf{x}}; \mathbf{w}_{\mathsf{r}}^{\mathsf{t}}) \cdot \big(C + \prod \mathfrak{p}(\mathsf{y}|\phi_{\mathbf{x}}; \mathbf{w}_{\mathsf{r}}^{\mathsf{t}})\big)^{-1} \cdot v \\
&\arg\max_{\mathbf{w}} \underbrace{\sum \log \mathfrak{p}(\mathsf{y}|\phi_{\mathbf{x}}; \mathbf{w}_{\mathsf{r}}^{\mathsf{t}})}_{\text{maximize data likelihood}} \underbrace{- \log\big(C + \prod \mathfrak{p}(\mathsf{y}|\phi_{\mathbf{x}}; \mathbf{w}_{\mathsf{r}}^{\mathsf{t}})\big)}_{\text{bias term}} + \log \mathfrak{v}^{\mathsf{t}} \\
&\overset{|\mathbb{D}| \to \infty}{\sim} \arg\max_{\mathbf{w}} \sum \log \mathfrak{p}(\mathsf{y}|\phi_{\mathbf{x}}; \mathbf{w}_{\mathsf{r}}^{\mathsf{t}}) \underbrace{- \log C' + \log \mathfrak{v}^{\mathsf{t}}.}_{\text{some constant to } \mathsf{Agent}_{\mathsf{r}}}
\end{aligned}$$

And see that the first term is the classical training objective of maximizing data likelihood as seen in remark E.3.4. While the second term is a bias term whereby the data likelihood under the $\mathsf{Agent}_{\mathsf{r}}$'s model appears again. Notably, for very large values of $|\mathbb{D}|$, we must have $\log\big(C + \prod \mathfrak{p}(\mathsf{y}|\phi_{\mathbf{x}}; \mathbf{w}_{\mathsf{r}}^{\mathsf{t}})\big) < 0$, as $0 < C, \mathsf{q} \ll 1$. Moreover, at large $|\mathbb{D}|$ and sufficiently large R, the term $\prod \mathfrak{p}(\mathsf{y}|\phi_{\mathbf{x}}; \mathbf{w}_{\mathsf{r}}^{\mathsf{t}})$ grows vanishingly small compared to the term C, so that the entire tail end of the expression is just a constant from the perspective of $\mathsf{Agent}_{\mathsf{r}}$. This suggests the reward $\mathfrak{v}_{\mathsf{r}}^{\mathsf{t}}$ does not affect the goal of maximizing data-likelihood in the limit. ■

Proof. for 2.5.2. We assume both $\mathsf{Agent}_{\mathsf{r}}$ and Agent_{s} expects to be one of the winners, that is with $\mathsf{r}, \mathsf{s} \in \{1, .., \mathsf{R}\}$. Then $\mathsf{Agent}_{\mathsf{r}}$'s winning wager split is of form:

$$\beta_{\mathsf{r}}^{*} = \frac{\prod \mathfrak{p}(\mathsf{y}|\phi_{\mathbf{x}}; \mathbf{w}_{\mathsf{r}}^{\mathsf{t}})}{e^{\mathsf{logprob}_{1}^{\mathsf{t}}} + \ldots + e^{\mathsf{logprob}_{\mathsf{R}}^{\mathsf{t}}}} \quad \beta_{\mathsf{r}} = \frac{\prod \mathfrak{p}(\mathsf{y}|\phi_{\mathbf{x}}; \mathbf{v}_{\mathsf{r}}^{\mathsf{t}})}{e^{\mathsf{logprob}_{1}^{\mathsf{t}}} + \ldots + e^{\mathsf{logprob}_{\mathsf{R}}^{\mathsf{t}}}},$$

and similarly for β_{s}^{*} and β_{s}. Then we have the payoff table:

$\mathsf{Agent}_{\mathsf{r}}$ \ Agent_{s}	$\mathbf{w}_{s}^{\mathsf{t}}$	$\mathbf{v}_{s}^{\mathsf{t}}$
$\mathbf{w}_{\mathsf{r}}^{\mathsf{t}}$	$\big(\beta_{\mathsf{r}}^{*}\mathfrak{v}^{\mathsf{t}},\ \beta_{\mathsf{s}}^{*}\mathfrak{v}^{\mathsf{t}}\big)$	$\big(\beta_{\mathsf{r}}^{*}\mathfrak{v}^{\mathsf{t}},\ \beta_{\mathsf{s}}\mathfrak{v}^{\mathsf{t}}\big)$
$\mathbf{v}_{\mathsf{r}}^{\mathsf{t}}$	$(\beta_{\mathsf{r}}\mathfrak{v}^{\mathsf{t}},\ \beta_{\mathsf{s}}^{*}\mathfrak{v}^{\mathsf{t}})$	$\big(\beta_{\mathsf{r}}\mathfrak{v}^{\mathsf{t}},\ \beta_{\mathsf{s}}\mathfrak{v}^{\mathsf{t}}\big)$

Figure C.2: A two player subgame, with winning $\mathsf{Agent}_{\mathsf{r}}$ competing against Agent_{s} for a portion of the reward $\mathfrak{v}^{\mathsf{t}}$. Note $\beta_{\mathsf{r}}^{*} > \beta_{\mathsf{r}}$, and $\beta_{\mathsf{s}}^{*} > \beta_{\mathsf{s}}$, given these two fixed permutations, any other permutation over the four β's are allowed.

Now referencing fig C.2, take the perspective of $\mathsf{Agent}_{\mathsf{r}}$ and consider how his reports rank w.r.t Agent_{s}:

1. $\beta_{\mathsf{r}}^* > \beta_{\mathsf{r}} > \beta_{\mathsf{s}}^* > \beta_{\mathsf{s}}$: clearly we have $\beta_{\mathsf{r}}^* \mathfrak{v}^{\mathsf{t}} > \beta_{\mathsf{r}} \mathfrak{v}^{\mathsf{t}}$, so $\mathsf{Agent}_{\mathsf{r}}$ reports $\mathbf{w}_{\mathsf{r}}^{\mathsf{t}}$.
2. $\beta_{\mathsf{r}}^* > \beta_{\mathsf{s}}^* > \beta_{\mathsf{r}} > \beta_{\mathsf{s}}$: it is still the case that $\beta_{\mathsf{r}}^* \mathfrak{v}^{\mathsf{t}} > \beta_{\mathsf{r}} \mathfrak{v}^{\mathsf{t}}$, so $\mathsf{Agent}_{\mathsf{r}}$ reports $\mathbf{w}_{\mathsf{r}}^{\mathsf{t}}$.
3. $\beta_{\mathsf{s}}^* > \beta_{\mathsf{r}}^* > \beta_{\mathsf{r}} > \beta_{\mathsf{s}}$: in this case if Agent_s reports $\mathbf{w}_s^{\mathsf{t}}$, then he takes a larger share of the rewards. But for $\mathsf{Agent}_{\mathsf{r}}$ we still have $\beta_{\mathsf{r}}^* \mathfrak{v}^{\mathsf{t}} > \beta_{\mathsf{r}} \mathfrak{v}^{\mathsf{t}}$, so $\mathsf{Agent}_{\mathsf{r}}$ reports $\mathbf{w}_{\mathsf{r}}^{\mathsf{t}}$.

A similar argument can proceed with locations of β_{s}. In sum, regardless of how the various β_{r} and β_{s} are ordered with respect to each other, reporting the truth $\mathbf{w}_{\mathsf{r}}^{\mathsf{t}}$ is the best response for $\mathsf{Agent}_{\mathsf{r}}$. Thus reporting the truth is a *pure strategy equilibrium* in the sub-game amongst winning Agents within each stage t. ■

A few caveats to the above lemmas follows:

1. it is only valid if we assume $\mathsf{cost}_{\mathsf{r}}^{\mathsf{t}}$ is constant and does not change regardless of how well the Agent trains his model. If the cost is some function of the Bregman divergence between the Agent's current model and his previous submission, i.e.: $v\mathsf{D}(\mathbf{w}_{\mathsf{r}}^{\mathsf{t}-1} || \mathbf{w}_{\mathsf{r}}^{\mathsf{t}})$. Then an Agent's optimal report is a convex combination of his last report and the best current report, as seen in eqn D.10. In this case, the Agent should maintain a distribution over acceptable $\mathbf{w}$'s, ranging over model parameters that is no worse than the last reported parameter $\mathbf{w}^{\mathsf{t}-1}$, and sample from this distribution. Thus reporting a better model than last stage is a *mixed strategy Nash Equilibrium*.
2. Note we assume the Agent does not know other Agents' report, hence the constant value C. This is a fair assumption as the submission stage is a closed one, and the stage t's models are not revealed until wagering time at $\mathsf{t}+1$.
3. In reality, each Agent submit generative model or data source, not a discriminative model $\mathbf{w}$. So they will report their honest $\boldsymbol{\theta}$ or $\mathbb{D}$ if they are confident that the corresponding classifier model trained by QAudit is a "fair" reflection of the original reports.

Remark C.1.1. We can also set C and $\mathfrak{v}^{\mathsf{t}}$ to be some random variable as they are unknowns from the perspective of the Agent. From a Bayesian perspective, this would make sense if we have some prior estimates of C and $\mathfrak{v}^{\mathsf{t}}$, and then can update them based on observations. But the Agent cannot observe other Agents' reports until the market opens on stage $\mathsf{t}+1$, and similarly for Auditors' wagers. Thus setting C and $\mathfrak{v}^{\mathsf{t}}$ as r.v. would be an unhelpful formality.

C.2 Agents-Auditors Sub-Game

Proof. for lemma 2.5.3. We prove the case where $\mathsf{R} \geq 2$. Given K Agents reporting classifier models $\mathbf{w}_{\mathsf{r}}^{\mathsf{t}}$ with convex cost function, and set of Auditors: $\{\mathsf{Auditor}_{\mathsf{i}}\}_{\mathsf{i}=1..\mathsf{N}}$. Now suppose for some stage t the two best models $\{\mathbf{w}_{\mathsf{r}}^{\mathsf{t}}, \mathbf{w}_s^{\mathsf{t}}\}$ are submitted by two Agents: $\{\mathsf{Agent}_{\mathsf{r}}, \mathsf{Agent}_s\}$. Recall in QAudit the Agents move first with the intention of competing against other Agents. And as C.2 shows, reporting their true $\mathbf{w}_{\mathsf{k}}^{\mathsf{t}}$ is each Agent's dominant strategy. At reward time, $\mathsf{QAudit}_{\mathfrak{v}}$ builds ensemble model $\mathbf{w}^{\mathsf{t}-1}$, and some $\mathbf{w}^{\mathsf{t}}$ is built from $\{\mathbf{w}_{\mathsf{r}}^{\mathsf{t}}, \mathbf{w}_s^{\mathsf{t}}\}$. They could rank in one of two ways:

$$\mathbf{E}_{\mathbb{D}\sim\mathfrak{p}}\left[\prod_{(\phi_{\mathbf{x}},\mathsf{y})\in\mathbb{D}} \mathfrak{p}(\mathsf{y}|\phi_{\mathbf{x}};\mathbf{w}^{\mathsf{t}-1})\right] > \text{ or } < \mathbf{E}_{\mathbb{D}\sim\mathfrak{p}}\left[\prod_{(\phi_{\mathbf{x}},\mathsf{y})\in\mathbb{D}} \mathfrak{p}(\mathsf{y}|\phi_{\mathbf{x}};\mathbf{w}^{\mathsf{t}})\right].$$

Here the expectation is with respect to the population distribution $\mathfrak{p}$ from which the test set $\mathbb{D}$ is drawn. Now if the Auditor expects the data likelihood under $\mathbf{w}^{\mathsf{t}-1}$ is greater than the current stage's ensemble model, then he would not wager at all. In the second case

where data likelihood under $\mathbf{w}^{\mathsf{t}}$ is greater, then we will show that 1) the Auditor is better off wagering on the winning Agents than some losing $\mathsf{Agent}_{\mathsf{k}}$, and 2) the Auditor should wager *more* on the better winning Agent measured by data likelihood. Suppose the $\mathsf{Auditor}_{\mathsf{i}}$ observes from the evaluation set $\mathbb{D}_{\mathsf{val}}$ $\exp(\mathsf{logprob}_{\mathsf{r}}^{\mathsf{t}}) > \exp(\mathsf{logprob}_{\mathsf{s}}^{\mathsf{t}})$, and some $\mathsf{Agent}_{\mathsf{k}}$ whose model is worse than both r and s. Furthermore, suppose the amount wagered on the losing Agents is $\mathfrak{v}^{\mathsf{t}}$, and the winning pot in stage $\mathsf{t}+1$ is $\mathfrak{v}^{\mathsf{t}+1}$. Then:

1. Auditor should not wager on $\mathsf{Agent}_{\mathsf{k}}$: wlog we select $\mathsf{Agent}_{\mathsf{r}}$. And it is clear from C.3 that $\mathsf{Auditor}_{\mathsf{i}}$'s dominant strategy is to select $\mathbf{w}_{\mathsf{r}}^{\mathsf{t}}$ over $\mathbf{w}_{\mathsf{k}}^{\mathsf{t}}$. From $\mathsf{Agent}_{\mathsf{r}}$'s perspective, in the event where his wager would have made a difference and placed $\mathsf{Agent}_{\mathsf{r}}$ into the top R slot, then $\mathsf{Agent}_{\mathsf{r}}$ is compensated with $\varrho_{\mathsf{r}}^{\mathsf{t}}$ instead of winning wager.

	$\mathsf{Auditor}_{\mathsf{i}}$	
	wager $\mathbf{w}_{\mathsf{r}}^{\mathsf{t}}$	**wager $\mathbf{w}_{\mathsf{k}}^{\mathsf{t}}$**
$\mathbf{w}_{\mathsf{r}}^{\mathsf{t}}$	$\left(\beta_{\mathsf{r}}\mathfrak{v}^{\mathsf{t}},\ \alpha_{\mathsf{r}}\mathfrak{v}_{l}^{\mathsf{t}+1}\right)$	$\left(\beta_{\mathsf{r}}\mathfrak{v}^{\mathsf{t}} \text{ or } \varrho_{\mathsf{r}}^{\mathsf{t}},\ 0\right)$

Figure C.3: A two player sub-game showing $\mathsf{Auditor}_{\mathsf{i}}$ should not defect from $\mathsf{Agent}_{\mathsf{r}}$. Here β_{r} is defined as above, and $\alpha_{\mathsf{r}} = \rho_{\mathsf{r}}^{\mathsf{t}} / \sum_{s=1}^{\mathsf{R}} \rho_{\mathsf{s}}^{\mathsf{t}}$.

2. $\mathsf{Auditor}_{\mathsf{i}}$ may benefit when wagering more on $\mathsf{Agent}_{\mathsf{r}}$'s model than that of Agent_{s}. The inequality here is the regret factor $\varrho_{\mathsf{i}}^{\mathsf{t}}$. In the event where $\mathsf{Agent}_{\mathsf{r}}$ does better both in hindsight and as stated on the evaluation set $\mathbb{D}_{\mathsf{val}}$, then $\mathsf{QAudit}_{\varrho\mathfrak{v}}$ allocates some portion of $\mathsf{Auditor}_{\mathsf{i}}$'s winnings back to $\mathsf{Agent}_{\mathsf{r}}$ for lost compensation.

	$\mathsf{Auditor}_{\mathsf{i}}$	
	wager $\mathbf{w}_{\mathsf{r}}^{\mathsf{t}}$	**wager $\mathbf{w}_{s}^{\mathsf{t}}$**
$\mathbf{w}_{\mathsf{r}}^{\mathsf{t}}$	$\left(\beta_{\mathsf{r}}\mathfrak{v}^{\mathsf{t}},\ \alpha_{\mathsf{r}}\mathfrak{v}_{l}^{\mathsf{t}+1}\right)$	$\left(\beta_{\mathsf{r}}\mathfrak{v}^{\mathsf{t}} + \varrho_{\mathsf{r}}^{\mathsf{t}} \text{ or } \varrho_{\mathsf{r}}^{\mathsf{t}},\ \alpha_{\mathsf{r}}'\mathfrak{v}_{l}^{\mathsf{t}+1} - \varrho_{\mathsf{i}}^{\mathsf{t}}\right)$

Figure C.4: A two player sub-game showing $\mathsf{Auditor}_{\mathsf{i}}$ should wager more on $\mathsf{Agent}_{\mathsf{r}}$ than Agent_{s} if data-likelihood is greater under $\mathbf{w}_{\mathsf{r}}^{\mathsf{t}}$. Here α_{r} and $\alpha_{\mathsf{r}}^{'}$ differ by the number of shares purchased for $\mathbf{w}_{\mathsf{r}}^{\mathsf{t}}$, with $\alpha_{\mathsf{r}} > \alpha_{\mathsf{r}}'$. The value $\mathfrak{v}_{l}^{\mathsf{t}+1}$ refers to the corresponding betting pool for slot l in stage $\mathsf{t}+1$.

Now consider $\mathsf{Auditor}_{\mathsf{i}}$'s total payout over both models $\mathbf{w}_{\mathsf{r}}^{\mathsf{t}}$ and $\mathbf{w}_{s}^{\mathsf{t}}$, for defecting from $\mathsf{Agent}_{\mathsf{r}}$ to Agent_{s}. Holding all other bids constant, we have:

$$\underbrace{\frac{\mathsf{s}_{\mathsf{r},\mathsf{i}}^{\mathsf{t}}}{\mathsf{s}_{\mathsf{r}}^{\mathsf{t}}}\mathfrak{v}_{\mathsf{a}}^{\mathsf{t}+1} + \frac{\mathsf{s}_{\mathsf{s},\mathsf{i}}^{\mathsf{t}}}{\mathsf{s}_{\mathsf{s}}^{\mathsf{t}}}\mathfrak{v}_{b}^{\mathsf{t}+1}}_{\text{lhs}} \overset{?}{\geq} \underbrace{\frac{\mathsf{s}_{\mathsf{r},\mathsf{i}}^{\mathsf{t}'}}{\mathsf{s}_{\mathsf{r}}^{\mathsf{t}'}}\mathfrak{v}_{b}^{\mathsf{t}+1} + \frac{\mathsf{s}_{\mathsf{s},\mathsf{i}}^{\mathsf{t}'}}{\mathsf{s}_{\mathsf{s}}^{\mathsf{t}'}}\mathfrak{v}_{\mathsf{a}}^{\mathsf{t}+1} - \varrho_{\mathsf{i}}^{\mathsf{t}}}_{\text{rhs}} \quad \text{where}$$

$$\varrho_{\mathsf{i}}^{\mathsf{t}} = \frac{\mathsf{s}_{i,r}^{\mathsf{t}}}{\mathsf{s}_{\mathsf{r}}^{\mathsf{t}}} \cdot \mathsf{D}(\mathbf{w}^{\mathsf{t}} || \mathbf{w}_{*}^{\mathsf{t}})$$

$$\mathsf{s}^{\mathsf{t}}_{\mathsf{r},\mathsf{i}} > \mathsf{s}^{\mathsf{t}'}_{\mathsf{r},\mathsf{i}}, \quad \mathsf{s}^{\mathsf{t}}_{\mathsf{s},\mathsf{i}} < \mathsf{s}^{\mathsf{t}'}_{\mathsf{s},\mathsf{i}} \quad \mathsf{s}^{\mathsf{t}}_{\mathsf{s}} > \mathsf{s}^{\mathsf{t}'}_{\mathsf{s}} \quad \mathsf{s}^{\mathsf{t}}_{\mathsf{r}} < \mathsf{s}^{\mathsf{t}'}_{\mathsf{r}}.$$

Where $\mathsf{s}^{\mathsf{t}}_{\mathsf{r},\mathsf{i}}$ is the number of shares $\mathsf{Auditor}_{\mathsf{i}}$ purchased when not defecting from $\mathsf{Agent}_{\mathsf{r}}$, $\mathsf{s}^{\mathsf{t}'}_{\mathsf{r},\mathsf{i}}$ is what the $\mathsf{Auditor}_{\mathsf{i}}$ purchased when he did defect, $\mathfrak{v}^{\mathsf{t}+1}_{\mathsf{a}}, \mathfrak{v}^{\mathsf{t}+1}_{b}$ are the total wagers for the respective top and 2nd slots in stage $\mathsf{t}+1$, and regret $\varrho^{\mathsf{t}}_{\mathsf{i}}$ is $\mathsf{Auditor}_{\mathsf{i}}$'s regret w.r.t $\mathbf{w}^{\mathsf{t}}_{*}$, or the best ensemble model.

Depending on the size of the Bregman divergence term $\mathsf{D}(\mathbf{w}^{\mathsf{t}}||\mathbf{w}^{\mathsf{t}}_{*})$, the *lhs* may or may not be greater than the *rhs*. Thus there are scenarios whereby $\mathsf{Auditor}_{\mathsf{i}}$ benefit from wagering more on Agent_{s} over the better ranked $\mathsf{Agent}_{\mathsf{r}}$ under log-likelihood of the evaluation set. This scenario could arise if the Auditor believes $\mathbf{w}^{\mathsf{t}}_{s}$ is sufficiently underpriced, and is willing to take the Bregman-regret hair-cut. In other words, Auditors should maintain a distribution over the top ranked Agents, perhaps informed by price of said Agents, and select from this distribution when wagering. All in all, under QAudit Auditor-Agent "collusion" is a *mixed strategy equilibrium.*

■

C.3 Auditors-Auditors Sub-Game within every Stage

Proof. for lemma 2.5.8 In this case we let $n = \mathsf{K}!$, and select some suitable order of $\mathfrak{s}$'s so we can define a PSA of form:

$$\mathsf{PSA} = \Big(\mathbb{S}_{\mathsf{K}} = \big\{\mathfrak{s}_1, .., \mathfrak{s}_{\mathsf{n}}\big\},\ \mathbb{A}_{\mathsf{K}} = \big\{(\mathfrak{s}_{\mathsf{r}}, \mathfrak{s}_{\mathsf{s}}) \in \mathbb{S}_{\mathsf{K}}, \mathsf{r} \leq \mathsf{s}\big\},\ \mathfrak{s}_o = \mathfrak{s}_1,\ \mathbb{S}^{*}_{\mathsf{K}} = \big\{\mathfrak{s}_{\mathsf{n}}\big\}\Big),$$

expressing a directed permutation graph with a single terminal state, or an "absorbing chain" [SW22, p. 24]. Let $\mathbb{S}_{\mathsf{K}}$ be a lexicographic ordering of the states, so that $\mathfrak{s}_1 = \big(1, 2, \ldots, \mathsf{K}\big)$, and $\mathfrak{s}_{\mathsf{n}} = \big(\mathsf{K}, \mathsf{K}-1, \ldots, 2, 1\big)$. In the worst case, we have $\mathfrak{s}_o = \mathfrak{s}_1$ and $\mathfrak{s}^{*}_{\mathsf{r}} = \mathfrak{s}_{\mathsf{n}}$, and the Auditors must cross the entire width of the PSA graph to find $\mathfrak{s}^{*}_{\mathsf{r}}$. For example if $\mathsf{K} = 3$, with $\mathfrak{s}_o = \big(1, 2, 3\big)$ and $\mathfrak{s}^{*}_{\mathsf{r}} = \big(3, 2, 1\big)$, then the longest path under neighbor transposition shuffle is: $\big[\big(1, 2, 3\big), \big(2, 1, 3\big), \big(2, 3, 1\big), \big(3, 2, 1\big)\big]$. And note that may be many paths of this length.

The expected time it takes for the Auditors to cross the graph by random walking is hitting time, denoted $\mathsf{h}_{1\mathsf{n}} := \mathsf{h}_{o*}$. This value can be computed by first expanding the process via a reverse random walk, and then bounding it from above. Recall AuditStrat endows the directed PSA graph with a set of transition probabilities:

$$\begin{aligned}
&\mathsf{state}_{\mathsf{n}}: && \mathsf{p}_{\mathsf{nn}} = 1\\
&\mathsf{state}_{\mathsf{n}-1}: && \mathsf{p}_{\mathsf{n}-1,\mathsf{n}-1} + \mathsf{p}_{\mathsf{a}} = 1\\
&\mathsf{state}_{\mathsf{n}-2}: && \mathsf{p}_{\mathsf{n}-2,\mathsf{n}-2} + \mathsf{p}_{\mathsf{a}} + \mathsf{p}_{\mathsf{n}-2,\mathsf{n}} = 1\\
&\mathsf{state}_{\mathsf{n}-3}: && \mathsf{p}_{\mathsf{n}-3,\mathsf{n}-3} + \mathsf{p}_{\mathsf{a}} + \mathsf{p}_{\mathsf{n}-3,\mathsf{n}-1} + \mathsf{p}_{\mathsf{n}-3,\mathsf{n}} = 1\\
&\ldots\\
&\mathsf{state}_{1}: && \mathsf{p}_{1,1} + \mathsf{p}_{\mathsf{a}} + \mathsf{p}_{1,3} + \ldots + \mathsf{p}_{1,\mathsf{n}} = 1.
\end{aligned}$$

Where every $0 \leq \mathsf{p}_{\mathsf{kk}} < 1$ for $\mathsf{k} = 1...\mathsf{n}$ is the probability of a self transition. While p_{a} is the probability of neighbor transposition. Note we assume that p_{a} is the same for every vertex in $\mathbb{S}_{\mathsf{K}}$. And $\mathsf{p}_{\mathsf{k},\mathsf{k}+l}$ for $l = 2...\mathsf{n}-\mathsf{k}$ is the probability of jumping l states in the PSA. Finally note we have $\mathsf{p}_{\mathsf{n}-2,1} = 1 - \mathsf{p}_{\mathsf{n}-2,\mathsf{n}-2} - \mathsf{p}_{\mathsf{a}}$, etc. In real life, this transition probability table may be constructed by collecting data from how Auditors wager. Here we can proceed with the reverse random walk using the variable names:

1. $\mathsf{h}_{\mathsf{n},\mathsf{n}}$: this is the sink, so the hitting time is 0 as the random walk has terminated.

2. $h_{n-1,n}$: this is the expected number of steps to transition from the penultimate state to the final state, written:

$$h_{n-1,n} = (1-p_a)(1+h_{n-1,n}) + p_a$$
$$h_{n-1,n} = \frac{1}{p_a}.$$

3. $h_{n-2,n}$: this is the expected number of steps to transition from the 2nd to last state to the final state:

$$h_{n-2,n} = p_{n-2,n-2}(1+h_{n-2,n}) + p_a(1+h_{n-1,n}) + 1\cdot(1-p_a-p_{n-2,n-2})$$
$$h_{n-2,n} = \frac{2}{1-p_{n-2,n-2}}.$$

4. $h_{n-3,n}$: as before we have:

$$\begin{aligned} h_{n-3,n} &= p_{n-3,n-3}(1+h_{n-3,n}) + p_a(1+h_{n-2,n}) + p_{n-3,n-1}(1+h_{n-1,n}) \\ &\quad + (1-p_a-p_{n-3,n-3}-p_{n-3,n}) \\ h_{n-3,n} &= \frac{2p_a^2 + p_{n-3,n-1}(1-p_{n-2,n-2}) + p_a(1-p_{n-2,n-2})}{p_a(1-p_{n-2,n-2})(1-p_{n-3,n-3})} \\ &\overset{(1)}{\leq} \frac{2p_a^2+p_a+p_{n-3,n-1}}{p_a(1-p_{n-2,n-2})(1-p_{n-3,n-3})}, \end{aligned}$$

where going into (1) we use $1-p \leq 1$.

5. $h_{n-4,n}$, we have:

$$\begin{aligned} h_{n-4,n} &= p_{n-4,n-4}(1+h_{n-4,n}) + p_a(1+h_{n-3,n}) + p_{n-4,n-2}(1+h_{n-2,n}) + p_{n-4,n-1}(1+h_{n-1,n}) \\ &\quad + 1 - p_a - p_{n-4,n-4} - p_{n-4,n-2} - p_{n-4,n-1} \\ h_{n-4,n}(1-p_{n-4,n-4}) &\leq \underbrace{\frac{p_a(2p_a^2+p_a+p_{n-3,n-1})}{p_a(1-p_{n-2,n-2})(1-p_{n-3,n-3})}}_{\text{a}} + \underbrace{\frac{2\cdot p_{n-4,n-2}}{1-p_{n-2,n-2}}}_{\text{B}} + \underbrace{\frac{p_{n-4,n-1}}{p_a}}_{\text{C}} + 1 \end{aligned}$$

In the numerator of A, B, C, and the 1 factor we have:

$$\begin{aligned} A: &\quad p_a(2p_a^2+p_a+p_{n-3,n-1}) && \leq 2p_a^3+p_a^2+p_a \\ B: &\quad 2\cdot p_{n-4,n-2}p_a(1-p_{n-3,n-3}) && \leq 2p_a \\ C: &\quad p_{n-4,n-1}(1-p_{n-2,n-2})(1-p_{n-3,n-3}) && \leq 1 \\ 1: &\quad p_a(1-p_{n-2,n-2})(1-p_{n-3,n-3}) && \leq p_a. \end{aligned}$$

So the sum of numerators is:

$$(2p_a^3+p_a^2+p_a) + 2p_a + p_a + 1 \leq 4(p_a^3+p_a^2+p_a+1).$$

Thus the hitting time is bounded from above with:

$$h_{n-4,n} \leq \frac{4(p_a^3+p_a^2+p_a+1)}{p_a(1-p_{n-2,n-2})(1-p_{n-3,n-3})(1-p_{n-4,n-4})}.$$

In general, we must have:

$$h_{n-(n-1),n} \leq \frac{(n)(p_a^n+p_a^{n-1}+\ldots+p_a+1)}{p_a\prod_{k=1}^{n-2}1-p_{k,k}} \implies$$

$$\mathsf{h}_{1,\mathsf{K}!} \leq \frac{\sum_{\mathsf{k}=0}^{\mathsf{K}!} \mathsf{p}_{\mathsf{a}}^{\mathsf{k}}}{\mathsf{p}_{\mathsf{a}} \cdot \prod_{\mathsf{k}=1}^{\mathsf{K}!-2}(1-\mathsf{p}_{\mathsf{k},\mathsf{k}})} \cdot \mathsf{K}!. \tag{C.2}$$

All in all, in the event where all the Auditors agree on the best permutation $\mathfrak{s}_{\mathsf{r}}^*$, the hitting time, or number of wagers required to permute $\mathfrak{s}_o$ onto $\mathfrak{s}_{\mathsf{K}!}$, is $\mathsf{poly}\big(\mathsf{K}!\big)$ as desired. Moreover, observe in the event where the Auditors buy enough shares on each wager to rearrange the permutation when it is not at $\mathfrak{s}_{\mathsf{r}}^*$, then we have the self transition probability $\mathsf{p}_{\mathsf{k},\mathsf{k}} \sim 0$, so that eqn (C.2) can be bounded "reasonably" from below by:

$$\mathsf{h}_{1,\mathsf{K}!} \geq \big(\mathsf{p}_{\mathsf{a}}^{-1} + \mathsf{p}_{\mathsf{a}}^{0} + \mathsf{p}_{\mathsf{a}}^{1} + \mathsf{p}_{\mathsf{a}}^{2} + \ldots + \mathsf{p}_{\mathsf{a}}^{\mathsf{K}!-2} + \mathsf{p}_{\mathsf{a}}^{\mathsf{K}!-1}\big) \cdot \mathsf{K}!, \tag{C.3}$$

where each p_{a} is the likelihood of transitioning to the next permutation in lexicographic order. Note the h.o.t are vanishing small, we have i.e $\mathsf{h}_{1,\mathsf{K}!} \geq \big(1 + \mathsf{p}_{\mathsf{a}}^{-1} + \mathsf{p}_{\mathsf{a}} + \mathsf{p}_{\mathsf{a}}^{2} + \mathsf{p}_{\mathsf{a}}^{3}\big) \cdot \mathsf{K}!$ ■

Proof. for lemma 2.5.10. In this case, we are working with the PSA expressing an undirected permutation graph:

$$\mathsf{PSA} = \Big(\mathbb{S}_{\mathsf{K}} = \big\{\mathfrak{s}_1, .., \mathfrak{s}_{\mathsf{K}!}\big\},\ \mathbb{A}_{\mathsf{K}} = \big\{(\mathfrak{s}_{\mathsf{r}}, \mathfrak{s}_{\mathsf{s}}) :\ \mathfrak{s}_{\mathsf{r}}, \mathfrak{s}_{\mathsf{s}} \in \mathbb{S}_{\mathsf{K}}\big\},\ \mathfrak{s}_o = *,\ \mathbb{S}_{\mathsf{K}}^* = \big\{\mathfrak{s}_{\mathsf{r}}^*, \ldots, \mathfrak{s}_{\mathsf{m}}^*\big\}\Big).$$

This undirected graph can be represented with the walk matrix $\mathbf{M}$. And by assumption it is the weighted sum of m directed graphs: $\mathbf{M} = \alpha_{\mathsf{r}}\mathbf{M}_{\mathsf{r}} + \ldots + \alpha_{\mathsf{m}}\mathbf{M}_{\mathsf{m}}$, rewritten:

$$\mathbf{M} = \alpha_{\mathsf{r}}\mathbf{M}_{\mathsf{r}} + (1-\alpha_{\mathsf{r}})\tilde{\mathbf{M}}, \ \text{ where } \tilde{\mathbf{M}} = \alpha_2\mathbf{M}_2 + \ldots + \alpha_{\mathsf{m}}\mathbf{M}_{\mathsf{m}}.$$

Now for this proof, we are taking the *subjective* perspective of QAudit whose public ranking is $\mathfrak{s}_{\mathsf{r}}^*$. So the protocol does not know other Auditors' private beliefs, it only observe that "sometimes" the PSA jumps to some random state $\mathfrak{s}_{\tau}^{\mathsf{t}}$ that it would not have taken. Thus we can simplify $\tilde{\mathbf{M}}$ to the uniform random walk matrix $\frac{\mathbf{e}\mathbf{e}^{\intercal}}{\mathsf{K}!}$, stating that with probability $1/K$, the PSA jumps to some random permutation state, written:

$$\mathbf{M} \sim \alpha_{\mathsf{r}}\mathbf{M}_{\mathsf{r}} + (1-\alpha_{\mathsf{r}})\frac{\mathbf{e}\mathbf{e}^{\intercal}}{\mathsf{K}!}. \tag{C.4}$$

Now the sensitivity of the stationary distribution $\mathbf{p}$ under $\mathbf{M}$ as a function of α_{r} can be computed. This argument is from [LM98, p. 362] and [CM99], and proceeds as follows:

$$\begin{aligned}
&\mathbf{p}^{\intercal} = \mathbf{p}\mathbf{M}_{\mathsf{r}} = \mathbf{p}\Big(\alpha_{\mathsf{r}}\mathbf{M}_{\mathsf{r}} + (1-\alpha_{\mathsf{r}})\frac{\mathbf{e}\mathbf{e}^{\intercal}}{\mathsf{K}!}\Big) && \Longrightarrow \\
&\mathbf{p}^{\intercal}\big(\mathbf{I} - \alpha_{\mathsf{r}}\mathbf{M}_{\mathsf{r}}\big) = \mathbf{p}\frac{\mathbf{e}\mathbf{e}^{\intercal}}{\mathsf{K}!} - \alpha_{\mathsf{r}}\mathbf{p}\frac{\mathbf{e}\mathbf{e}^{\intercal}}{\mathsf{K}!} && \Longrightarrow \\
&\frac{\partial}{\partial\alpha_{\mathsf{r}}}\Big(\mathbf{p}^{\intercal}\big(\mathbf{I} - \alpha_{\mathsf{r}}\mathbf{M}_{\mathsf{r}}\big)\Big) = \frac{\partial}{\partial\alpha_{\mathsf{r}}}\Big(\mathbf{p}\frac{\mathbf{e}\mathbf{e}^{\intercal}}{\mathsf{K}!} - \alpha_{\mathsf{r}}\mathbf{p}\frac{\mathbf{e}\mathbf{e}^{\intercal}}{\mathsf{K}!}\Big) && \Longrightarrow \\
&\frac{\partial\mathbf{p}}{\partial\alpha_{\mathsf{r}}}^{\intercal}\big(\mathbf{I} - \alpha_{\mathsf{r}}\mathbf{M}_{\mathsf{r}}\big) + \mathbf{p}\frac{\partial}{\partial\alpha_{\mathsf{r}}}\big(\mathbf{I} - \alpha_{\mathsf{r}}\mathbf{M}_{\mathsf{r}}\big) = \frac{\partial}{\partial\alpha_{\mathsf{r}}}\Big(\mathbf{p}\frac{\mathbf{e}\mathbf{e}^{\intercal}}{\mathsf{K}!}\Big) - \frac{\partial}{\partial\alpha_{\mathsf{r}}}\Big(\alpha_{\mathsf{r}}\mathbf{p}\frac{\mathbf{e}\mathbf{e}^{\intercal}}{\mathsf{K}!}\Big) && \Longrightarrow \\
&\frac{\partial\mathbf{p}}{\partial\alpha_{\mathsf{r}}}^{\intercal}\big(\mathbf{I} - \alpha_{\mathsf{r}}\mathbf{M}_{\mathsf{r}}\big) - \mathbf{p}\mathbf{M}_{\mathsf{r}} = 0 - \mathbf{p}\frac{\mathbf{e}\mathbf{e}^{\intercal}}{\mathsf{K}!} && \Longrightarrow \\
&\frac{\partial\mathbf{p}}{\partial\alpha_{\mathsf{r}}}^{\intercal} = \mathbf{p}\Big(\mathbf{M}_{\mathsf{r}} - \frac{\mathbf{e}\mathbf{e}^{\intercal}}{\mathsf{K}!}\Big)\Big(\mathbf{I} - \alpha_{\mathsf{r}}\mathbf{M}_{\mathsf{r}}\Big)^{-1}.
\end{aligned}$$

Where going into the last line, we know the inverse of $\mathbf{I} - \alpha_{\mathsf{r}}\mathbf{M}_{\mathsf{r}}$ must exist as it is non-singular. Now we bound the last line, consider some value at index i in $\frac{\partial\mathbf{p}}{\partial\alpha_{\mathsf{r}}}$:

$$\Big(\frac{\partial\mathbf{p}}{\partial\alpha_{\mathsf{r}}}^{\intercal}\Big)_{\mathsf{i}} = \mathbf{p}\Big(\mathbf{M}_{\mathsf{r}} - \frac{\mathbf{e}\mathbf{e}^{\intercal}}{\mathsf{K}!}\Big)\Big(\mathbf{I} - \alpha_{\mathsf{r}}\mathbf{M}_{\mathsf{r}}\Big)^{-1}\mathbf{b}_{\mathsf{i}},$$

where $\mathbf{b}_{\mathsf{i}}$ is the n-bit vector with value 1 at index i, and zeros everywhere else. Using the identity $|\mathbf{x}^\intercal \mathbf{y}| \leq \|\mathbf{x}\|_1 \frac{\mathsf{y}_{max}-\mathsf{y}_{min}}{2}$, and setting $\mathbf{y} = \big(\mathbf{I} - \alpha_{\mathsf{r}}\mathbf{M}_{\mathsf{r}}\big)^{-1}\mathbf{b}_{\mathsf{i}}$, we have:

$$\begin{aligned}\left|\left(\frac{\partial \mathbf{p}^\intercal}{\partial \alpha_{\mathsf{r}}}\right)_{\mathsf{i}}\right| &\leq \left\|\mathbf{p}\left(\mathbf{M}_{\mathsf{r}} - \frac{\mathbf{e}\mathbf{e}^\intercal}{\mathsf{K!}}\right)\right\|_1 \cdot \frac{\mathsf{y}_{max}-\mathsf{y}_{min}}{2} \\ &\overset{(1)}{\leq} 2 \cdot \frac{\mathsf{y}_{max}-\mathsf{y}_{min}}{2} \\ &\leq \mathsf{y}_{max} - \mathsf{y}_{min},\end{aligned}$$

where (1) follows from $\left\|\mathbf{p}\big(\mathbf{M}_{\mathsf{r}} - \frac{\mathbf{e}\mathbf{e}^\intercal}{\mathsf{K!}}\big)\right\|_1 \leq 2$ as the product is akin to a probability vector. Next bound y_{max} by:

$$\mathsf{y}_{max} \overset{(1)}{\leq} \|(\mathbf{I} - \alpha_{\mathsf{r}}\mathbf{M}_{\mathsf{r}})^{-1}\mathbf{b}_{\mathsf{i}}\|_\infty \overset{(2)}{\leq} \|(\mathbf{I} - \alpha_{\mathsf{r}}\mathbf{M}_{\mathsf{r}})^{-1}\mathbf{e}\|_\infty \overset{(3)}{\leq} \frac{1}{1-\alpha_{\mathsf{r}}},$$

where (1) follows from the fact that the infinity norm is defined with $\|\mathbf{x}\|_\infty = max_{\mathsf{i}}|x_{\mathsf{i}}|$. (2) follows from the fact that $\mathbf{y} = \big(\mathbf{I} - \alpha_{\mathsf{r}}\mathbf{M}_{\mathsf{r}}\big)^{-1}\mathbf{b}_{\mathsf{i}} \leq \big(\mathbf{I} - \alpha_{\mathsf{r}}\mathbf{M}_{\mathsf{r}}\big)^{-1}\mathbf{e}$, as $\mathbf{e}$ is the all ones vector. And (3) follows from evaluating $(\mathbf{I} - \alpha_{\mathsf{r}}\mathbf{M}_{\mathsf{r}})\mathbf{e}$. For example in the $\mathsf{n} \times \mathsf{n}$ case we have:

$$\begin{aligned}(\mathbf{I} - \alpha_{\mathsf{r}}\mathbf{M}_{\mathsf{r}})\mathbf{e} &= \left(\begin{bmatrix} 1 & \ddots & 0 \\ & \ddots & \\ 0 & \ddots & 1 \end{bmatrix} - \begin{bmatrix} \alpha_{\mathsf{r}}\mathsf{p}_{11} & \dots & \alpha_{\mathsf{r}}\mathsf{p}_{1\mathsf{n}} \\ \vdots & \ddots & \vdots \\ \alpha_{\mathsf{r}}\mathsf{p}_{n1} & \dots & \alpha_{\mathsf{r}}\mathsf{p}_{\mathsf{nn}} \end{bmatrix}\right)(1,\dots,1)^\intercal \\ &= \Big(1 - \alpha_{\mathsf{r}}\mathsf{p}_{11} - \dots - \alpha_{\mathsf{r}}\mathsf{p}_{1\mathsf{n}}, \dots, 1 - \alpha_{\mathsf{r}}\mathsf{p}_{n1} - \dots - \alpha_{\mathsf{r}}\mathsf{p}_{\mathsf{nn}}\Big)^\intercal.\end{aligned}$$

Note every row sum yields: $1 - \alpha_{\mathsf{r}}\mathsf{p}_{\mathsf{r}1} - \dots - \alpha_{\mathsf{r}}\mathsf{p}_{\mathsf{rn}} = 1 - \alpha_{\mathsf{r}}(\sum_{\mathsf{i}=1}^{\mathsf{n}} \mathsf{p}_{\mathsf{ri}}) = 1 - \alpha_{\mathsf{r}}$. So the maximum value can be bounded with:

$$\big((\mathbf{I} - \alpha_{\mathsf{r}}\mathbf{M}_{\mathsf{r}})\mathbf{e}\big)_{\mathsf{i}} = 1 - \alpha_{\mathsf{r}} \implies \big((\mathbf{I} - \alpha_{\mathsf{r}}\mathbf{M}_{\mathsf{r}})^{-1}\mathbf{e}\big)_{\mathsf{i}} = \frac{1}{1-\alpha_{\mathsf{r}}}.$$

Similarly, we know $\mathsf{y}_{min} \geq 0$. Putting it all together and we have:

$$\left|\left(\frac{\partial \mathbf{p}^\intercal}{\partial \alpha_{\mathsf{r}}}\right)_{\mathsf{i}}\right| \leq \mathsf{y}_{max} \leq \frac{1}{1-\alpha_{\mathsf{r}}} \overset{(1)}{\implies} \frac{\partial \mathbf{p}^\intercal}{\partial \alpha_{\mathsf{r}}} \leq \frac{2}{1-\alpha_{\mathsf{r}}},$$

where (1) follows from the fact that $\mathbf{p}$ is a probability vector. Next we discretize this statement by perturbing α_{r} with some $\Delta\alpha_{\mathsf{r}}$ so that: $0 \leq \alpha_{\mathsf{r}} + \Delta\alpha_{\mathsf{r}} < 1$, and define the original and perturbed stationary distribution with:

$$\mathbf{p}_1{}^\intercal = \mathbf{p}_1\left(\alpha_{\mathsf{r}}\mathbf{M}_{\mathsf{r}} + (1-\alpha_{\mathsf{r}})\frac{\mathbf{e}\mathbf{e}^\intercal}{\mathsf{K!}}\right) \qquad \mathbf{p}_2{}^\intercal = \mathbf{p}_2\left((\alpha_{\mathsf{r}} + \Delta\alpha_{\mathsf{r}})\mathbf{M}_{\mathsf{r}} + (1-\alpha_{\mathsf{r}} - \Delta\alpha_{\mathsf{r}})\frac{\mathbf{e}\mathbf{e}^\intercal}{\mathsf{K!}}\right).$$

Then by common sense and [IW06, p. 6]:

$$\|\mathbf{p}_1 - \mathbf{p}_2\|_1 \leq \frac{2}{1-\alpha_{\mathsf{r}}}|\Delta\alpha_{\mathsf{r}}|.$$

Finally, referencing eqn 2.5, set $\alpha_{\mathsf{r}} = 1-\epsilon$, and write $\mathbf{p}^\intercal = \mathbf{p}\big((\alpha_{\mathsf{r}} - \Delta\alpha_{\mathsf{r}})\mathbf{M}_{\mathsf{r}} + (1-\alpha_{\mathsf{r}}+\Delta\alpha_{\mathsf{r}})\frac{\mathbf{e}\mathbf{e}^\intercal}{\mathsf{K!}}\big)$, with $0 \leq \Delta\alpha_{\mathsf{r}} < 1-\epsilon$. Then we have:

$$\|\tilde{\boldsymbol{\delta}}_{\mathsf{r}} - \mathbf{p}\|_1 \leq \frac{2}{1-\alpha_{\mathsf{r}}}|\Delta\alpha_{\mathsf{r}}| = \frac{2|\Delta\alpha_{\mathsf{r}}|}{\epsilon},$$

as eqn 2.6 states. ■

Proof. for lemma 2.5.11. Consider a partition over Auditors where some portion p have private belief $\mathfrak{q}_{\mathsf{r}}^{\mathsf{t}}$, and $(1-\mathsf{p})$ believes in $\mathfrak{q}_{s}^{\mathsf{t}}$:

$$\begin{cases} \mathsf{p} & \text{play suggested action } \mathfrak{s}_{\mathsf{r}}^{*} \\ 1-\mathsf{p} & \text{hold private belief } \mathfrak{q}_{s}^{\mathsf{t}} \text{ that induces } \mathfrak{s}_{\mathsf{s}}. \end{cases} \qquad 0 < \mathsf{p} < 1.$$

Given these parameters, we can simplify the PSA graph into just three states so that $\mathfrak{s}_c$ is the "super-state" composed of every non-r,s states: $\mathfrak{s}_c := \left\{\mathfrak{s}_{\mathsf{i}} \,:\, \mathsf{i}=1...\mathsf{K}, \mathsf{i}\neq \mathsf{r},\mathsf{s}\right\}$. Call this $\mathfrak{s}_c$ "no-man's land." Then the PSA is of form:

$$\mathsf{PSA} = \left(\mathbb{S}_{\mathsf{K}} = \left\{\mathfrak{s}_{\mathsf{r}}^{*}, \mathfrak{s}_{\mathsf{s}}, \mathfrak{s}_c\right\}, \quad \mathbb{A}_{\mathsf{K}} = \left\{(\mathfrak{s}_{\mathsf{i}}, \mathfrak{s}_{\mathsf{j}}) \,:\, \mathfrak{s}_{\mathsf{i}}, \mathfrak{s}_{\mathsf{j}} \in \mathbb{S}_{\mathsf{K}}\right\}, \ \mathfrak{s}_o = *, \quad \mathbb{S}_{\mathsf{K}}^{*} = \left\{\mathfrak{s}_{\mathsf{s}}, \mathfrak{s}_{\mathsf{r}}^{*}\right\}\right).$$

Notice the initial state $\mathfrak{s}_o$ is arbitrary as there is a unique steady state distribution over states regardless of initialization point. Additionally, we assume:

1. QAudit samples Auditors with uniform randomness, so that the likelihood of sampling any Auditor holding belief $\mathfrak{s}_{\mathsf{r}}^{*}$ is their population proportion p.

2. if the PSA is at the intermediate state $\mathfrak{s}_c$, then the PSA will transition with likelihood:

$$\mathfrak{p}\left(X_{\tau+1} | X_\tau = \mathfrak{s}_c\right) = \begin{cases} 2\mathsf{q} & \text{remain in state } \mathfrak{s}_c \\ \mathsf{p}-\mathsf{q} & \text{transition to state } \mathfrak{s}_{\mathsf{r}}^{*} \\ \mathbf{1}-\mathsf{p}-\mathsf{q} & \text{transition to state } \mathfrak{s}_{\mathsf{s}} \end{cases} \qquad 0 \leq \mathsf{q} < \frac{1}{2}.$$

3. The Auditors jump from state $\mathfrak{s}_{\mathsf{r}}^{*}$ to $\mathfrak{s}_{\mathsf{s}}$ with probability ϵ, and vice versa.

All in all, we can define state transition matrix $\mathbf{M}$ with steady state probability vector $\mathbf{p} = \left[\mathfrak{p}(\mathfrak{s}_c^{*}), \mathfrak{p}(\mathfrak{s}_{\mathsf{r}}), \mathfrak{p}(\mathfrak{s}_{\mathsf{s}})\right]$, [1] so that $\mathbf{p}^{\intercal}\mathbf{e} = 1$, and $\mathbf{p} = \mathbf{p}\mathbf{M}$ per eqn (B.1):

$$\left[\mathfrak{p}(\mathfrak{s}_c^{*}), \mathfrak{p}(\mathfrak{s}_{\mathsf{r}}), \mathfrak{p}(\mathfrak{s}_{\mathsf{s}})\right]^{\intercal} = \left[\mathfrak{p}(\mathfrak{s}_c^{*}), \mathfrak{p}(\mathfrak{s}_{\mathsf{r}}), \mathfrak{p}(\mathfrak{s}_{\mathsf{s}})\right] \begin{bmatrix} 2\mathsf{q} & \mathsf{p}-\mathsf{q} & 1-\mathsf{p}-\mathsf{q} \\ 1-\mathsf{p}-\epsilon & \mathsf{p} & \epsilon \\ \mathsf{p}-\epsilon & \epsilon & 1-\mathsf{p} \end{bmatrix},$$

and note that $\mathbf{M}$ is row-stochastic. Solving this expression, we find a solution set of steady state distributions that is a function of p and q:

1. (case $\mathsf{p} = \frac{1}{2}$): in this case we also have: $0 < \epsilon < \frac{1}{2}$, and $0 < \mathsf{q} \leq \frac{1}{3}$:

$$\mathfrak{p}(\mathfrak{s}_c) = \frac{1-2\epsilon}{3-4\mathsf{q}-2\epsilon} \quad \mathfrak{p}(\mathfrak{s}_{\mathsf{r}}^{*}) = \frac{1-2\mathsf{q}}{3-4\mathsf{q}-2\epsilon} \quad \mathfrak{p}(\mathfrak{s}_{\mathsf{s}}) = \frac{1-2\mathsf{q}}{3-4\mathsf{q}-2\epsilon}.$$

   Unsurprisingly, when the Auditors' opinion are divided evenly, then the likelihood of finding $\mathfrak{s}_{\mathsf{r}}^{*}$ is just as likely as finding $\mathfrak{s}_{\mathsf{s}}$. Moreover for $\epsilon > 1/3$, the likelihood of landing in no-man's land is greater than finding either one of the two private permutations. At $\epsilon = 1/3$, the distribution over the states is uniform.

2. (case $\frac{1}{2} < \mathsf{p} < 1$, $0 < \epsilon < \mathsf{p}$, $0 < \mathsf{q} \leq \frac{1}{3}$):

$$\mathfrak{p}(\mathfrak{s}_c) = \frac{\mathsf{p}-\mathsf{p}^2-\epsilon^2}{\mathsf{p}^2-\mathsf{p}-\mathsf{q}(2\epsilon+1)-\epsilon^2+\epsilon+1} \quad \mathfrak{p}(\mathfrak{s}_{\mathsf{r}}^{*}) = \frac{\mathsf{p}^2-\mathsf{p}(\mathsf{q}+\epsilon)-\mathsf{q}\epsilon+\epsilon}{\mathsf{p}^2-\mathsf{p}-\mathsf{q}(2\epsilon+1)-\epsilon^2+\epsilon+1}$$
$$\mathfrak{p}(\mathfrak{s}_{\mathsf{s}}) = \frac{\mathsf{p}^2+\mathsf{p}(\mathsf{q}+\epsilon-2)-\mathsf{q}(1+\epsilon)+1}{\mathsf{p}^2-\mathsf{p}-\mathsf{q}(2\epsilon+1)-\epsilon^2+\epsilon+1}$$

   This is the case where more Auditors believe in $\mathfrak{s}_{\mathsf{r}}^{*}$ than $\mathfrak{s}_{\mathsf{s}}$. The solution space can be partitioned in according to additional conditions:

[1] Note $\mathbf{p}$ is not the same as the price vector ρ^{t}, which expresses the likelihood that any $\mathsf{Agent}_{\mathsf{K}}$ will finish first on stage t.

a) If we require the likelihood of $\mathfrak{p}(\mathfrak{s}_\mathsf{r}^*) > \mathfrak{p}(\mathfrak{s}_c)$, then:

$$\mathfrak{p}(\mathfrak{s}_\mathsf{r}^*) > \mathfrak{p}(\mathfrak{s}_c) \implies$$
$$\underbrace{\frac{1}{2} < \mathsf{p} \leq \frac{2}{3},\ 0 < \mathsf{q} < 2\mathsf{p} - 1,\ 0 < \epsilon < \mathsf{p}}_{(1)} \text{ or}$$
$$\underbrace{\frac{1}{2} < \mathsf{p} \leq \frac{2}{3},\ 2\mathsf{p} - 1 < \mathsf{q} \leq \frac{1}{3},\ \frac{1}{2}\big(\sqrt{6\mathsf{pq} - 7\mathsf{p}^2 + 2\mathsf{p} + \mathsf{p}^2 - 2\mathsf{q} + 1} + \mathsf{p} + \mathsf{q} - 1\big) < \epsilon < \mathsf{p}}_{(2)} \text{ or}$$
$$\underbrace{\frac{2}{3} < \mathsf{p} \leq 1,\ 0 < \mathsf{q} \leq \frac{1}{3},\ 0 < \epsilon < p.}_{(3)}$$

In conditions (1) and (2) the proportion p of Auditors holding belief $\mathfrak{s}_\mathsf{r}^*$ may be less than 2/3, provided the probability of transitioning out of no-man's land is better than 1/3. In condition (3), whereby more than 2/3 of the Auditors hold belief $\mathfrak{s}_\mathsf{r}^*$, then no additional constraints is placed on q and ϵ above the assumptions of the current solution.

b) In general, it is useful to define an arbitrary minimum values of $\mathfrak{p}(\mathfrak{s}_\mathsf{r}^*)$, and determine what the value of p, q, and ϵ must be to achieve the desired threshold. For example with numerical constraints $\mathfrak{p}(\mathfrak{s}_\mathsf{r}^*) \geq 0.8$, we have many solutions:

$$\mathfrak{p}(\mathfrak{s}_\mathsf{r}^*) > 0.8 \implies$$
$$\underbrace{0.615 < \mathsf{p} \leq 0.8,\ \frac{4 - 5\mathsf{p}}{4 - 2\mathsf{p}} < \mathsf{q} < \frac{1}{3},\ l_\epsilon < \epsilon < \mathsf{p}}_{(1)}$$
$$\underbrace{0.8 < \mathsf{p} \leq 0.829,\ 0 < \mathsf{q} < \frac{1}{3},\ l_\epsilon < \epsilon < \mathsf{p},}_{(2)} \text{ or}$$
$$\underbrace{0.829 < \mathsf{p} \leq 0.840,\ 0 < \mathsf{q} < \frac{\mathsf{p}^2 + 4\mathsf{p} - 4}{5\mathsf{p} - 4},\ l_\epsilon < \epsilon < \mathsf{p},}_{(3)}$$
$$\text{where } l_\epsilon = \frac{1}{8}\big(\sqrt{9\mathsf{p}^2 + 50\mathsf{pq} - 74\mathsf{p} + 9\mathsf{q}^2 - 58\mathsf{p} + 65} + 5\mathsf{p} - 3\mathsf{q} - 1\big).$$

Notably, in (1) we see it is possible to achieve more than 80% likelihood of landing on $\mathfrak{s}_\mathsf{r}^*$ when only 62% of the Auditors hold this opinion, provided the likelihood of remaining in $\mathfrak{s}_c$ no-man's land is less than 2/3, and ϵ is larger than some lower bound l_{eps}.

Next the proportion p is allowed to be larger than 0.8, and the likelihood of staying in no-man's land is also allowed to be greater.

3. (case $\frac{1}{2} < \mathsf{p} < \frac{1-\mathsf{q}}{2\mathsf{q}}$, $0 < \epsilon < \mathsf{p}$, $\frac{1}{3} < \mathsf{q} \leq \frac{1}{2}$), alternatively we can have $0 < \epsilon < \frac{1}{2}\big(\sqrt{4\mathsf{p}^2 - 4\mathsf{p} + 4\mathsf{q}^2 - 8\mathsf{q} + 5} - 2\mathsf{q} + 1\big)$:

$$\mathfrak{p}(\mathfrak{s}_c) = \frac{\mathsf{p} - \mathsf{p}^2 - \epsilon^2}{\mathsf{p}^2 - \mathsf{p} - \mathsf{q}(2\epsilon + 1) - \epsilon^2 + \epsilon + 1} \quad \mathfrak{p}(\mathfrak{s}_\mathsf{r}^*) = \frac{\mathsf{p}^2 - \mathsf{p}(\mathsf{q} + \epsilon) - \mathsf{q}\epsilon + \epsilon}{\mathsf{p}^2 - \mathsf{p} - \mathsf{q}(2\epsilon + 1) - \epsilon^2 + \epsilon + 1}$$
$$\mathfrak{p}(\mathfrak{s}_\mathsf{s}) = \frac{\mathsf{p}^2 + \mathsf{p}(\mathsf{q} + \epsilon - 2) - \mathsf{q}(\epsilon + 1) + 1}{\mathsf{p}^2 - \mathsf{p} - \mathsf{q}(2\epsilon + 1) - \epsilon^2 + \epsilon + 1}.$$

Similar to the case above, we can place conditions on the desired value of $\mathfrak{p}(\mathfrak{s}_\mathsf{r}^*)$:

$$\mathfrak{p}(\mathfrak{s}_\mathsf{r}^*) > 0.8 \implies$$

$$\underbrace{0.5 < \mathsf{p} \leq 0.615,\ \frac{5\mathsf{p}-4}{2\mathsf{p}-4} < \mathsf{q} < \frac{1}{2\mathsf{p}+1},\ l_\epsilon < \epsilon < \mathsf{p}}_{(1)}$$

$$\underbrace{0.615 < \mathsf{p} \leq 0.840,\ \frac{1}{3} < \mathsf{q} < \frac{1}{2\mathsf{p}+1},\ l_\epsilon < \epsilon < \mathsf{p}}_{(2)} \quad \text{or}$$

$$\underbrace{0.842 < \mathsf{p} \leq 0.957,\ \frac{1}{3} < \mathsf{q} < \frac{1}{2\mathsf{p}+1},\ l_\epsilon < \epsilon < \mathsf{p},}_{(3)}$$

$$\text{where } l_\epsilon = \frac{1}{8}\big(\sqrt{9\mathsf{p}^2 + 50\mathsf{pq} - 74\mathsf{p} + 9\mathsf{q}^2 - 58\mathsf{p} + 65} + 5\mathsf{p} - 3\mathsf{q} - 1\big).$$

We can achieve the desired outcome of $\mathfrak{p}(\mathfrak{s}_\mathsf{r}^*)$ with a slightly majority value in p as seen in case (1), provided the likelihood of staying in $\mathfrak{s}_c$ is lesser than 2/3. [2]

Now since the label of $\mathfrak{s}_\mathsf{r}^*$ is arbitrary, the same analysis applies to $\mathfrak{s}_\mathsf{s}$ if that is the prompted ranking. All in all, this means that if the proportion of Auditors who hold private belief $\mathfrak{s}_\mathsf{r}^*$ is greater than 50%, then the likelihood of finding the solution $\mathfrak{s}_\mathsf{r}^*$ can be greater than 50% under some very mild conditions. ■

C.4 Fundamental Theorem across Stages

Proof. for theorem 2.6.10. In this proof, we select any $\mathsf{Auditor}_\mathsf{i}$, and fix $\mathfrak{v}_\mathsf{t}^\mathsf{i}$ across the the Auditors so that $\mathfrak{v}_\mathsf{t}^\mathsf{i} = \mathfrak{v}_\mathsf{t}$. And note we have switched the position of the index from the conventional notation so that:

$$\mathfrak{v}_\mathsf{t}^\mathsf{i} := \mathfrak{v}_\mathsf{i}^\mathsf{t}.$$

This avoid confusion later. Next pick some interval over stages $[\mathsf{t}^*, \mathsf{t}^* + \Delta\mathsf{t}^*]$, where 1) t^* is the last regime change point, and 2) the future regime change stage $\mathsf{t}^* + \Delta\mathsf{t}^*$ can be either odd or even. Then we can compute the present value of a sequence of wagers over the stage interval. We make two assumptions before proceeding:

1. because of the Dutch auction factor, the price of $\mathfrak{mint}_\mathsf{k}^\mathsf{t}$ shares increase on every stage, and *assuming* the Auditors are wagering in a tit-for-tat manner, then we can write:

$$\mathfrak{v}_{\mathsf{t}+1} = \mathfrak{v}_\mathsf{t} + \Delta\mathfrak{v}_\mathsf{t}, \quad \text{where } \Delta\mathfrak{v}_\mathsf{t} > 0, \quad \text{and } \mathsf{t} \in [\mathsf{t}^*, \mathsf{t}_e^*]. \tag{C.5}$$

   Stating that the $\mathsf{Auditor}_\mathsf{i}$ and $\mathsf{Auditor}_j$ are dutifully marking up each other's wagers on every stage in accordance to Dutch auction factor $\mathsf{v}_\rho^\mathsf{t}$. *And* by assumption $\mathsf{Auditor}_\mathsf{i}$ is winning in every stage t.

2. Suppose $\mathsf{t}^* + \Delta\mathsf{t}^*$ will take value according to:

$$\mathsf{t}^* + \Delta\mathsf{t}^* := \begin{cases} \mathsf{even} & \text{with probability } \mathsf{p} \\ \mathsf{odd} & \text{with probability } 1 - \mathsf{p}. \end{cases}$$

3. For each scenario above, $\mathsf{Auditor}_\mathsf{i}$ joins Odds and Evens clique with probability:

$$\mathsf{Auditor}_\mathsf{i}\ \mathsf{plays} := \begin{cases} \mathsf{Evens} & \text{with probability } \mathsf{q} \\ \mathsf{Odds} & \text{with probability } 1 - \mathsf{q}. \end{cases}$$

   Note that p and q are independent as Auditors do not know a-priori whether $\mathsf{t}^* + \Delta\mathsf{t}^*$ is odd or even.

[2] All of the arithmetic here is done by wolframalpha.com.

Then we have the game tree whereby for any $\mathsf{Auditor_i}$, his expected value for playing is:

$$\mathbf{E}[\mathsf{x}] = \mathsf{p}(\mathsf{q}\mathsf{v}_{ee} + (1-\mathsf{q})\mathsf{v}_{eo}) + (1-\mathsf{p})(\mathsf{q}\mathsf{v}_{oe} + (1-\mathsf{q})\mathsf{v}_{oo}).$$

Where v_{ee} is the discounted value of the game to $\mathsf{Auditor_i}$ if $\mathsf{t}^* + \Delta\mathsf{t}^*$ is even, and he joins the Evens clique; v_{eo} is the discounted value if $\mathsf{t}^* + \Delta\mathsf{t}^*$ is even, and the $\mathsf{Auditor_i}$ joins the Odds clique, and so on. We may compute x as follows:

1. Let $\mathsf{t}^*_e = \mathsf{t}^* + \Delta\mathsf{t}^*$ be even, then if $\mathsf{Auditor_i}$ enter the game on the Odds stages, his profits are:

$$\begin{aligned}
\mathsf{v}_{eo} &\overset{(1)}{=} -\Big(\underbrace{\mathfrak{v}_{\mathsf{t}^*+1}\cdot(1+\curlyvee)^{-1} + \mathfrak{v}_{\mathsf{t}^*+3}\cdot(1+\curlyvee)^{-2} + \ldots + \mathfrak{v}_{\mathsf{t}^*_e-1}\cdot(1+\curlyvee)^{-(\mathsf{t}^*_e-1)}}_{\mathsf{Auditor_i}\text{'s wagers on odd stages}}\Big) \\
&\quad + \Big(\underbrace{\mathfrak{v}_{\mathsf{t}^*+2}\cdot(1+\curlyvee)^{-1} + \mathfrak{v}_{\mathsf{t}^*+4}\cdot(1+\curlyvee)^{-4} + \ldots + \mathfrak{v}_{\mathsf{t}^*_e-2}\cdot(1+\curlyvee)^{-(\mathsf{t}^*_e-2)}}_{\mathsf{Auditor}_j\text{'s wagers on even stages}}\Big) \\
&\overset{(2)}{\geq} \left(-\mathfrak{v}_{\mathsf{t}^*+1} + \underbrace{\mathfrak{v}_{\mathsf{t}^*+1} + \Delta\mathfrak{v}_{\mathsf{t}^*+1}}_{\mathsf{Auditor}_j\text{'s wager } \mathfrak{v}_{\mathsf{t}^*+2}}\right)\frac{1}{(1+\curlyvee)^{\mathsf{t}^*+1}} + .. \\
&\quad + \left(-\mathfrak{v}_{\mathsf{t}^*_e-3} + \underbrace{\mathfrak{v}_{\mathsf{t}^*_e-3} + \Delta\mathfrak{v}_{\mathsf{t}^*_e-3}}_{\mathsf{Auditor}_j\text{'s wager } \mathfrak{v}_{\mathsf{t}^*_e-2}}\right)\frac{1}{(1+\curlyvee)^{\mathsf{t}^*_e-3}} - \mathfrak{v}_{\mathsf{t}^*_e-1}\frac{1}{(1+\curlyvee)^{\mathsf{t}^*_e-1}}.
\end{aligned}$$

Now we define the profit hurdle with:

$$\begin{aligned}
&\overset{(3)}{\Longrightarrow} \quad \underbrace{\frac{\Delta\mathfrak{v}_{\mathsf{t}^*+1}}{(1+\curlyvee)^{\mathsf{t}^*+1}} + \ldots + \frac{\Delta\mathfrak{v}_{\mathsf{t}^*_e-3}}{(1+\curlyvee)^{\mathsf{t}^*_e-3}}}_{\mathsf{Auditor_i}\text{'s revenue assuming he wins every round}} > \underbrace{\frac{\mathfrak{v}_{\mathsf{t}^*_e-1}}{(1+\curlyvee)^{\mathsf{t}^*_e-1}}}_{\mathsf{Auditor_i}\text{'s spending}} \\
&\overset{(4)}{\Longrightarrow} \quad \frac{\Delta\mathfrak{v}}{(1+\curlyvee)^{\mathsf{t}^*+1}} + \ldots + \frac{\Delta\mathfrak{v}}{(1+\curlyvee)^{\mathsf{t}^*_e-3}} > \frac{\mathfrak{v}_{\mathsf{t}^*_e-1}}{(1+\curlyvee)^{\mathsf{t}^*_e-1}}
\end{aligned} \tag{C.6}$$

Where we reason as follows:

a) going into (1) we note that the $\mathsf{Evens\ Auditor}$ would not wager on round $\mathsf{t}^*_e = \mathsf{t}^* + \Delta\mathsf{t}^*$, since he can see that the models have been reset. So the Odds wager one more round than the Evens.

b) Going into (2), we substitute eqn (C.5) and rewrite the $\mathsf{Evens\ Auditor}_j$'s wager as a function of the $\mathsf{Odds\ Auditor_i}$'s wager on the previous stage. And use the inequality:

$$-\frac{\mathfrak{v}_{\mathsf{t}^*+1}}{(1+\curlyvee)^{\mathsf{t}^*+1}} + \frac{\mathfrak{v}_{\mathsf{t}^*+1} + \Delta\mathfrak{v}_{\mathsf{t}^*+1}}{(1+\curlyvee)^{\mathsf{t}^*+2}} \geq \frac{-\mathfrak{v}_{\mathsf{t}^*+1} + \mathfrak{v}_{\mathsf{t}^*+1} + \Delta\mathfrak{v}_{\mathsf{t}^*+1}}{(1+\curlyvee)^{\mathsf{t}^*+1}},$$

to pull the discount factor backward one step on $\mathfrak{v}_{\mathsf{t}^*+2} = \mathfrak{v}_{\mathsf{t}^*+1} + \Delta\mathfrak{v}_{\mathsf{t}^*+1}$. And observe that if the model improvement between consecutive stages t is sufficiently small, then $\curlyvee$ is also small, so the two values above are approximately equal.

c) Going into (3), we assume $\mathsf{Auditor_i}$ wins every round, and compute his discounted revenue. This assumption is fair as the previous theorems showed that if the $\mathsf{Auditors}$ just follow the QAudit prompt, then they will find the best permutation over Agents. Now because of 1) the sequential parimutuel payout structure, and 2) the Dutch auction price ascending factor $\mathsf{v}^{\mathsf{t}}_{\rho}$, there is some earnings $\Delta\mathfrak{v}_{\mathsf{t}}$ for every round t. That is except the penultimate round $\mathsf{t}^*_e - 1$, when the $\mathsf{Evens\ Auditor}_j$

refuse to wager due to regime change. Note as $(1+\curlywedge)^{\mathsf{t}}$ is monotonically decreasing with:

$$(1+\curlywedge)^{-(\mathsf{t}_e^*-1)} < (1+\curlywedge)^{-(\mathsf{t}_e^*-2)} < \ldots < (1+\curlywedge)^{-(\mathsf{t}^*+1)},$$

we know that $(1+\curlywedge)^{-(\mathsf{t}_e^*-1)}$ must be the smallest value.

d) Going into (4), we know by construction of $\mathsf{v}_\rho^{\mathsf{t}}$ that $\Delta\mathfrak{v}_{\mathsf{t}}$ must be the same for every t on a per share basis assuming both initialization price vector v_ρ^o and purchasing volume are constant across stages. Then we know $\Delta\mathfrak{v}_{\mathsf{t}^*+l} = \Delta\mathfrak{v}$ for every $l = 1, \ldots, \mathsf{t}_e^* - 1$.

e) All in all, we see that the drivers of profit risk are:

i. *Regime change risk* $\curlywedge$: this is a function of the underlying improvement in models over consecutive stages.

ii. *Price at* $\mathsf{t}_e^* - 1$: the ratio $\mathfrak{v}_{\mathsf{t}_e^*-1}/\Delta\mathfrak{v}$ determines the amount of loss $\mathsf{Auditor_i}$ takes at the moment of regime change. In particular, the break-even point is:

$$\Delta\mathsf{t}^*\Delta\mathfrak{v} = \mathfrak{v}_{\mathsf{t}_e^*-1}.$$

This suggests that if regime change is frequent, the Dutch auction factor $\mathsf{v}_\rho^{\mathsf{t}}$ must be aggressive.

2. Alternatively, suppose t_e^* is even as before, and $\mathsf{Auditor_i}$ choose to wager with the Evens, then his profits are:

$$\begin{aligned}
\mathsf{v}_{ee} &\overset{(1)}{=} \underbrace{+\left(\mathfrak{v}_{\mathsf{t}^*+1}\cdot(1+\curlywedge)^{-1} + \mathfrak{v}_{\mathsf{t}^*+3}\cdot(1+\curlywedge)^{-2} + \ldots + \mathfrak{v}_{\mathsf{t}_e^*-1}\cdot(1+\curlywedge)^{-(\mathsf{t}_e^*-1)}\right)}_{\mathsf{Auditor}_j\text{'s wagers on odd stages}} \\
&\quad \underbrace{-\left(\mathfrak{v}_{\mathsf{t}^*+2}\cdot(1+\curlywedge)^{-2} + \mathfrak{v}_{\mathsf{t}^*+4}\cdot(1+\curlywedge)^{-4} + \ldots + \mathfrak{v}_{\mathsf{t}_e^*-2}\cdot(1+\curlywedge)^{-(\mathsf{t}_e^*-2)}\right)}_{\mathsf{Auditor_i}\text{'s wagers on even stages}} \\
&\overset{(2)}{\geq} \frac{-\mathfrak{v}_{\mathsf{t}^*+2}+\mathfrak{v}_{\mathsf{t}^*+2}+\Delta\mathfrak{v}_{\mathsf{t}^*+2}}{(1+\curlywedge)^{\mathsf{t}^*+2}} + \ldots + \frac{-\mathfrak{v}_{\mathsf{t}_e^*-2}+\mathfrak{v}_{\mathsf{t}_e^*-2}+\Delta\mathfrak{v}_{\mathsf{t}_e^*-2}}{(1+\curlywedge)^{\mathsf{t}_e^*-2}} \\
&\overset{(3)}{\Longrightarrow} \frac{\Delta\mathfrak{v}_{\mathsf{t}^*+2}}{(1+\curlywedge)^{\mathsf{t}^*+2}} + \frac{\Delta\mathfrak{v}_{\mathsf{t}^*+4}}{(1+\curlywedge)^{\mathsf{t}^*+4}} + \ldots + \frac{\Delta\mathfrak{v}_{\mathsf{t}_e^*-2}}{(1+\curlywedge)^{\mathsf{t}_e^*-2}} \geq 0 \\
&\overset{(4)}{\Longrightarrow} \frac{\Delta\mathfrak{v}}{(1+\curlywedge)^{\mathsf{t}^*+2}} + \frac{\Delta\mathfrak{v}}{(1+\curlywedge)^{\mathsf{t}^*+4}} + \ldots + \frac{\Delta\mathfrak{v}}{(1+\curlywedge)^{\mathsf{t}_e^*-2}} \geq 0.
\end{aligned} \tag{C.7}$$

The reasoning are similar to the Odds case:

a) going into (1) we simply flipped the sign and the index i, j from above. Note in this case, the Evens does not start playing until t^*+2, and plays until stage t_e^*-2, as they observe the regime change in stage t_e^*.

b) Going into (2), we substitute eqn (C.5) and proceed with the same inequality as before.

c) Going into (3), we note that the profit is always at least zero, provided that the Evens $\mathsf{Auditor_i}$'s ensemble model always defeats the previous stages' models. This means if $\mathsf{Auditor_i}$ choose correctly and plays the Evens, then he never loses money *if* t_e^* is even.

d) Going into (4), we use the same reasoning around $\Delta\mathfrak{v}$ as before.

3. Reasoning just so, and let $\mathsf{t}_o^* = \mathsf{t}^* + \Delta\mathsf{t}^*$ be odd, so we have interval $[\mathsf{t}^*, \mathsf{t}_o^*]$. Then we find v_{oo}'s expected value is no less than zero:

$$\frac{\Delta\mathfrak{v}}{(1+\curlyvee)^{\mathsf{t}^*+1}} + \frac{\Delta\mathfrak{v}}{(1+\curlyvee)^{\mathsf{t}^*+3}} + \ldots + \frac{\Delta\mathfrak{v}}{(1+\curlyvee)^{\mathsf{t}_o^*-2}} \geq 0. \tag{C.8}$$

4. And similarly t_o^* is odd as before, then v_{oe} must meet hurdle:

$$\frac{\Delta\mathfrak{v}}{(1+\curlyvee)^{\mathsf{t}^*+2}} + \frac{\Delta\mathfrak{v}}{(1+\curlyvee)^{\mathsf{t}^*+4}} + \ldots + \frac{\Delta\mathfrak{v}}{(1+\curlyvee)^{\mathsf{t}_o^*-3}} \geq \frac{\mathfrak{v}_{\mathsf{t}_o^*-1}}{(1+\curlyvee)^{\mathsf{t}_o^*-1}}. \tag{C.9}$$

Consequentially, $\mathsf{Auditor_i}$'s expected value discounted by regime risk is:

$$\begin{aligned}\mathbf{E}[\mathsf{x}] = \mathsf{p}\cdot\Delta\mathfrak{v}\cdot\left(\frac{1-\mathsf{q}}{(1+\curlyvee)^{\mathsf{t}^*+1}} + \frac{\mathsf{q}}{(1+\curlyvee)^{\mathsf{t}^*+2}} + \ldots + \frac{1-\mathsf{q}}{(1+\curlyvee)^{\mathsf{t}_e^*-3}} + \frac{\mathsf{q}}{(1+\curlyvee)^{\mathsf{t}_e^*-2}} - \frac{(1-\mathsf{q})\mathfrak{v}_{\mathsf{t}_e^*-1}/\Delta\mathfrak{v}}{(1+\curlyvee)^{\mathsf{t}_e^*-1}}\right)\\ + (1-\mathsf{p})\cdot\Delta\mathfrak{v}\cdot\left(\frac{\mathsf{q}}{(1+\curlyvee)^{\mathsf{t}^*+1}} + \frac{1-\mathsf{q}}{(1+\curlyvee)^{\mathsf{t}^*+2}} + \ldots + \frac{\mathsf{q}}{(1+\curlyvee)^{\mathsf{t}_o^*-3}} + \frac{1-\mathsf{q}}{(1+\curlyvee)^{\mathsf{t}_o^*-2}} - \frac{\mathsf{q}\mathfrak{v}_{\mathsf{t}_o^*-1}/\Delta\mathfrak{v}}{(1+\curlyvee)^{\mathsf{t}_o^*-1}}\right).\end{aligned}$$

So we have:

$$\begin{aligned}\mathbf{E}[\mathsf{x}] = &\underbrace{\mathsf{p}\Delta\mathfrak{v}\sum_{\mathsf{t}=1}^{\mathsf{t}_e^*}\frac{(1-\mathsf{q})(\mathsf{t}\bmod 2)+\mathsf{q}(1+\mathsf{t}\bmod 2)}{(1+\curlyvee)^{\mathsf{t}^*+\mathsf{t}}} + (1-\mathsf{p})\Delta\mathfrak{v}\sum_{\mathsf{t}=1}^{\mathsf{t}_o^*}\frac{\mathsf{q}(\mathsf{t}\bmod 2)+(1-\mathsf{q})(1+\mathsf{t}\bmod 2)}{(1+\curlyvee)^{\mathsf{t}^*+\mathsf{t}}}}_{\mathsf{Auditor_i}\text{'s earnings until regime change}}\\ &\underbrace{-\frac{(1-\mathsf{q})\mathfrak{v}_{\mathsf{t}_e^*-1}}{(1+\curlyvee)^{\mathsf{t}_e^*-1}} - \frac{\mathsf{q}\mathfrak{v}_{\mathsf{t}_o^*-1}}{(1+\curlyvee)^{\mathsf{t}_o^*-1}}}_{\mathsf{Auditor_i}\text{'s loss on regime change}}\end{aligned} \tag{C.10}$$

We can leave this expression in unreduced form as $\mathsf{t}_o^* \neq \mathsf{t}_e^*$. And observe that the regime change risk $-(1+\curlyvee)^{-(\mathsf{t}_e^*-1)}$ and $-(1+\curlyvee)^{-(\mathsf{t}_o^*-1)}$ are vanishingly small, if $\mathsf{t}_e^* - \mathsf{t}^*$ and $\mathsf{t}_o^* - \mathsf{t}^*$ are sufficiently large. Now in the special case where:

1. $\mathsf{t}_o^* = \mathsf{t}_e^* + 1$,
2. $\mathsf{q} = \frac{1}{2}$, so $\mathsf{q} = 1 - \mathsf{q}$,

then we can simplify the expression further. Consider the sum:

$$\mathsf{p}\cdot\frac{1-\mathsf{q}}{(1+\curlyvee)^{\mathsf{t}^*+1}} + (1-\mathsf{p})\cdot\frac{\mathsf{q}}{(1+\curlyvee)^{\mathsf{t}^*+1}} = \frac{\mathsf{pq}}{(1+\curlyvee)^{\mathsf{t}^*+1}}.$$

Proceeding as above for every term in C.10, we have:

$$\begin{aligned}\frac{1}{\Delta\mathfrak{v}}\cdot\mathbf{E}[\mathsf{x}] &= \frac{\mathsf{pq}}{(1+\curlyvee)^{\mathsf{t}^*+1}} + \frac{\mathsf{pq}}{(1+\curlyvee)^{\mathsf{t}^*+2}} + \ldots + \frac{\mathsf{pq}}{(1+\curlyvee)^{\mathsf{t}_e^*-3}} + \frac{\mathsf{pq}}{(1+\curlyvee)^{\mathsf{t}_e^*-2}} + \frac{\mathsf{pq}}{(1+\curlyvee)^{\mathsf{t}_e^*-1}}\\ &\quad - \frac{\mathfrak{v}_{\mathsf{t}_e^*-1}}{\Delta\mathfrak{v}}\cdot\frac{\mathsf{pq}}{(1+\curlyvee)^{\mathsf{t}_e^*-1}} - \frac{\mathfrak{v}_{\mathsf{t}_e^*}}{\Delta\mathfrak{v}}\cdot\frac{(1-\mathsf{p})\mathsf{q}}{(1+\curlyvee)^{\mathsf{t}_e^*}}\\ &\overset{(1)}{\leq} -\frac{\mathfrak{v}_{\mathsf{t}_e^*}}{\Delta\mathfrak{v}}\cdot\left(\frac{1}{(1+\curlyvee)^{\mathsf{t}_e^*-1}} + \frac{1}{(1+\curlyvee)^{\mathsf{t}_e^*}}\right) + \sum_{\mathsf{t}=1}^{\mathsf{t}_e^*-1}\frac{\mathsf{pq}}{(1+\curlyvee)^{\mathsf{t}^*+\mathsf{t}}}\\ &\overset{(2)}{\leq} -\frac{2\mathfrak{v}_{\mathsf{t}_e^*}/\Delta\mathfrak{v}}{(1+\curlyvee)^{\mathsf{t}_e^*-1}} + \sum_{\mathsf{t}=1}^{\mathsf{t}_e^*-1}\frac{\mathsf{pq}}{(1+\curlyvee)^{\mathsf{t}^*+\mathsf{t}}}\end{aligned}$$

$$
\begin{aligned}
&\overset{(3)}{\Longrightarrow} \sum_{\mathsf{t}=1}^{\mathsf{t}_e^*-1} \frac{\mathsf{pq}}{\big(1+\curlywedge\big)^{\mathsf{t}^*+\mathsf{t}}} \;\geq\; \frac{2\mathfrak{v}_{\mathsf{t}_e^*}/\Delta\mathfrak{v}}{\big(1+\curlywedge\big)^{\mathsf{t}_e^*-1}} \\
&\overset{(4)}{\Longrightarrow} \sum_{\mathsf{t}=1}^{\mathsf{t}_e^*-1} \frac{\mathsf{p}}{\big(1+\curlywedge\big)^{\mathsf{t}^*+\mathsf{t}}} \;\geq\; \frac{4\mathfrak{v}_{\mathsf{t}_e^*}}{\Delta\mathfrak{v}} \frac{1}{\big(1+\curlywedge\big)^{\mathsf{t}_e^*-1}} \\
&\overset{(5)}{\Longrightarrow} \\
&\mathfrak{v}_{\mathsf{t}_e^*} \leq \Delta\mathfrak{v}\frac{\mathsf{p}}{4}\sum_{\mathsf{t}=2}^{\mathsf{t}_e^*}(1+\curlywedge)^{\mathsf{t}_e^*-\mathsf{t}} \quad \text{or} \quad \frac{\mathfrak{v}_{\mathsf{t}_e^*}}{\Delta\mathfrak{v}} \leq \frac{\mathsf{p}}{4}\sum_{\mathsf{t}=2}^{\mathsf{t}_e^*}(1+\curlywedge)^{\mathsf{t}_e^*-\mathsf{t}} \text{ or} \\
&\Delta\mathfrak{v} \geq \frac{4\mathfrak{v}_{\mathsf{t}_e^*}}{\mathsf{p}} \cdot \frac{1}{\sum_{\mathsf{t}=2}^{\mathsf{t}_e^*}(1+\curlywedge)^{\mathsf{t}_e^*-\mathsf{t}}} \overset{(6)}{\Longrightarrow} \frac{8\mathfrak{v}_{\mathsf{t}_e^*}}{\sum_{\mathsf{t}=2}^{\mathsf{t}_e^*}(1+\curlywedge)^{\mathsf{t}_e^*-\mathsf{t}}} \text{ if } \mathsf{p}=\frac{1}{2}.
\end{aligned} \tag{C.11}
$$

We reason as follows:

1. going into (1) we use the fact that $\mathsf{p}, 1-\mathsf{p}, \mathsf{q} \leq 1$, and $\mathfrak{v}_{\mathsf{t}_e^*-1} \leq \mathfrak{v}_{\mathsf{t}_e^*}$ given price markup $\Delta\mathfrak{v}_{\mathsf{t}}$ on every stage t.
2. Going into (2), we observe:
$$\frac{2}{\big(1+\curlywedge\big)^{\mathsf{t}_e^*-1}} \geq \frac{1}{\big(1+\curlywedge\big)^{\mathsf{t}_e^*}} + \frac{1}{\big(1+\curlywedge\big)^{\mathsf{t}_e^*-1}}.$$
3. Going into (3), we note that $\mathsf{pq} = \frac{\mathsf{p}}{2}$.
4. Going into $(4, 5)$, we reduce every term in the sum with:
$$\frac{(1+\curlywedge)^{\mathsf{t}_e^*-1}}{(1+\curlywedge)^{l}} = (1+\curlywedge)^{\mathsf{t}_e^*-1-l}$$
for every $l = 1..\mathsf{t}_e^*-1$. And then set $\mathbf{E}[\mathsf{x}] \geq 0$ to solve for the conditions on $\mathfrak{v}_{\mathsf{t}_e^*}$ and $\Delta\mathfrak{v}$.
5. Going into (6), we set $\mathsf{p} = \frac{1}{2}$ to arrive at eqn 2.13

In fact, we can reduce C.11 further by setting $\mathfrak{v}_{\mathsf{t}_e^*} = \epsilon + (\mathsf{t}_e^*-1)\Delta\mathfrak{v}$, where $\mathfrak{v}_{\mathsf{t}_o^*} := \epsilon$. And then we have:

$$
\begin{aligned}
&\Delta\mathfrak{v} \geq \frac{8(\epsilon+(\mathsf{t}_e^*-1)\Delta\mathfrak{v})}{\sum_{\mathsf{t}=2}^{\mathsf{t}_e^*}(1+\curlywedge)^{\mathsf{t}_e^*-\mathsf{t}}} \implies \\
&\Delta\mathfrak{v} \geq \frac{8\epsilon}{\big(\sum_{\mathsf{t}=2}^{\mathsf{t}_e^*}(1+\curlywedge)^{\mathsf{t}_e^*-\mathsf{t}}\big)-8(\mathsf{t}_e^*-1)} = \frac{\epsilon}{\max\Big[1, \big(\sum_{\mathsf{t}=2}^{\mathsf{t}_e^*}(1+\curlywedge)^{\mathsf{t}_e^*-\mathsf{t}}\big)-8(\mathsf{t}_e^*-1)\Big]}.
\end{aligned}
$$

Where going into the equal sign, we set $8\epsilon = \epsilon$. A similar analysis can be done for $\mathsf{t}_e^* = \mathsf{t}_o^*+1$. All in all, we can see that as $\Delta\mathsf{t}^*$ increases, it becomes more feasible to satisfy conditions C.11, for every value of $\curlywedge$. For example, for $\Delta\mathsf{t}^* = 100$, $\epsilon = 1$, and $\curlywedge = 0.05$, we have $\Delta\mathfrak{v} \geq 0.000515$. Thus QAudit must balance Dutch auction ascending factor $\mathsf{v}_\rho^{\mathsf{t}}$ with the frequency of regime change to maintain Auditor retention. Now given conditions C.11 are met, and as the choice of interval $[\mathsf{t}^*, \mathsf{t}^*+\Delta\mathsf{t}^*]$ is arbitrary, we know that the multi-stage QAudit game has positive expected value for every $\mathsf{Auditor}_\mathsf{i}$'s coin toss decision to play Odds or Evens. ■

C.5 FeaAudit

Proof. for proposition 3.3.3. Suppose some $\mathsf{Cohort}^{\mathsf{n}}$ has N Witnesses. And suppose every Witness is called, then their testimony before the PA is an bit string of length N. The set of all possible N-bit strings form a boolean-cube with 2^{N} vertices, where the edges correspond to flipping a single bit. We conceptualize a random experiment as follows:

1. There is a true label N-bit string:

$$\mathbf{b} = \big[\mathsf{y}_1, \ldots, \mathsf{y}_{\mathsf{t}}, \ldots, \mathsf{y}_{\mathsf{N}}\big],$$

   corresponding to some vertex on the hypercube. Where the elements of $\mathbf{b}$ are arranged in chronological order w.r.t the while loop of FeaAudit.

2. At some stage t, the Witness named $\mathsf{Wit}_{\mathsf{t}}^{\mathsf{n}}$ is called to testify, his true preference is y_{t} as seen in $\mathbf{b}$.

3. The FeaAudit protocol then flips this y with probability $1-\mathsf{p}$, as seen in eqn 3.2. So the probability of flipping a bit is:

$$\mathfrak{p}(\tilde{\mathsf{y}} = \mathsf{Yes} \mid \mathsf{y} = \mathsf{No}) + \mathfrak{p}(\tilde{\mathsf{y}} = \mathsf{No} \mid \mathsf{y} = \mathsf{Yes}) = \frac{1}{2}(1-\mathsf{p}) + \frac{1}{2}(1-\mathsf{p}) = 1-\mathsf{p}.$$

   Critically, the flips are independent. This creates a N-bit string where the masked bit $\tilde{\mathsf{y}}_{\mathsf{t}}$ is flipped with likelihood $1-\mathsf{p}$.

4. After all N Witnesses have testified, we have the corrupted string:

$$\tilde{\mathbf{b}} = \big[\tilde{\mathsf{y}}_1, \ldots, \tilde{\mathsf{y}}_{\mathsf{t}}, \ldots, \tilde{\mathsf{y}}_{\mathsf{N}}\big], \quad \text{where } \tilde{\mathsf{y}}_{\mathsf{t}} = \mathsf{y}_{\mathsf{t}} \text{ with probability } \mathsf{p}.$$

Now we have to determine 1) the *expected* distance between $\tilde{\mathbf{b}}$ and $\mathbf{b}$ measured in edge traversals; and 2) the likelihood that $\tilde{\mathbf{b}}$ and $\mathbf{b}$ differ by more than some amount so as to render the collected data "too noisy" to use. Denoting distance with: $X = \|\tilde{\mathbf{b}} - \mathbf{b}\|_1$, so that the distance is x when x bits have been flipped. Then the expected value of this distance is:

$$\begin{aligned}
\mathbf{E}\big[X\big] &= \sum_{\mathsf{x}=0}^{\mathsf{N}} \mathsf{x}\binom{\mathsf{N}}{\mathsf{x}}\mathsf{p}^{\mathsf{N}-\mathsf{x}}(1-\mathsf{p})^{\mathsf{x}} \overset{(1)}{=} \sum_{\mathsf{x}=0}^{\mathsf{N}} \mathsf{N}\binom{\mathsf{N}-1}{\mathsf{x}-1}\mathsf{p}^{\mathsf{N}-\mathsf{x}}(1-\mathsf{p})^{\mathsf{x}} \\
&= \mathsf{N}\sum_{\mathsf{x}=0}^{\mathsf{N}-1} \mathsf{N}\binom{\mathsf{N}-1}{\mathsf{x}}\mathsf{p}^{\mathsf{N}-\mathsf{x}-1}(1-\mathsf{p})^{\mathsf{x}+1} \overset{(2)}{=} \mathsf{N}(1-\mathsf{p})\sum_{\mathsf{x}=0}^{\mathsf{N}-1}\binom{\mathsf{N}-1}{\mathsf{x}}\mathsf{p}^{\mathsf{N}-\mathsf{x}-1}(1-\mathsf{p})^{\mathsf{x}} \\
&= \mathsf{N}(1-\mathsf{p}).
\end{aligned}$$

Where going into (1), we use $\mathsf{x}\binom{\mathsf{N}}{\mathsf{x}} = \mathsf{N}\binom{\mathsf{N}-1}{\mathsf{x}-1}$, going into (2) the sum is 1. And the expectation is w.r.t the coin tosses of the first $\mathsf{coin}_{\mathsf{p}}$, and the second fair coin with $\mathfrak{p}(\mathsf{Head}) = \frac{1}{2}$. Now we require this distance to be less than some d where $\mathsf{d} = 0, 1, 2, \ldots$:

$$\mathsf{N}(1-\mathsf{p}) \leq \mathsf{d} \implies \mathsf{p} \geq 1 - \frac{\mathsf{d}}{\mathsf{N}}.$$

Finally using Chernoff bound of def B.1.3, we can determine the likelihood that X is more than $(1 \pm \delta)$ away from $\mathbf{E}[X] = \mathsf{d}$. For every $\delta \in (0, 1)$:

$$\mathfrak{p}\big(X \geq (1+\delta)\mathsf{d}\big) \leq \exp\Big(-\frac{\delta^2\mathsf{d}}{3}\Big).$$

Stating the likelihood of encountering any $\tilde{\mathbf{b}}$ generated by FeaAudit that strays too far from $\mathbf{E}[X]$ decays exponentially in d. So the overall N-bit string summarizing any Cohort's preference reflects the actual value $\mathbf{b}$ if the PA sets $\mathsf{coin}_{\mathsf{p}}$ so that $\mathsf{p} \geq 1 - \frac{\mathsf{d}}{\mathsf{N}}$. ■

Proof. for proposition 3.3.6. Observe that the PA's $\mathsf{coin}^{\mathsf{t}}_{\mathsf{q}}$ has bias $(\mathsf{q}_l, \mathsf{q}_u) = (\frac{1}{3}, \frac{2}{3})$. Referencing eqn 3.4, we have:

$$\mathsf{q}_l^2 \leq \mathfrak{p}(\mathsf{Head}) \leq \mathsf{q}_u^2, \quad \text{for every } \mathsf{t},$$

And by 3.5 the eunuch's survival time at court is between $\mathsf{t}^* \in [2.25, 9]$ stages. Now suppose VA_k enters the eunuch position with initial wealth $\mathfrak{v}_\mathsf{k}$, and setting the condition so that he can *expect* to leave the eunuch seat with more wealth than when he started. Then we have this condition on the earning sequence:

$$\begin{aligned}
&(1-\xi)\left(\mathfrak{v}_\mathsf{k} + \frac{\xi}{2}\sum_{\mathsf{t}=0}^{\mathsf{t}^*}\mathfrak{v}_\mathsf{i}\right) \geq \mathfrak{v}_\mathsf{k} \ \text{ where } \mathsf{i} \in [1, \ldots] \qquad \text{(C.12)}\\
&\implies \mathfrak{v}_\mathsf{k} \geq \frac{1-\xi}{2}\sum_{\mathsf{t}=0}^{\mathsf{t}^*}\mathfrak{v}_\mathsf{i} \overset{(1)}{\implies} \mathfrak{v}_\mathsf{k} \geq \frac{1-\xi}{2}\left(\frac{1}{\mathsf{q}^2}+1\right)\mathfrak{v}\\
&\implies \xi \geq 1 - 2\frac{\mathfrak{v}_\mathsf{k}}{\mathfrak{v}}\frac{\mathsf{q}^2}{1+\mathsf{q}^2} \overset{(2)}{\implies} \xi \geq 1 - \frac{2\mathsf{q}^2}{1+\mathsf{q}^2}\\
&\implies \xi > 0.384 \ \text{ if } \mathsf{q} = \frac{2}{3}, \ \text{ or } \xi < 0.8 \ \text{ if } \mathsf{q} = \frac{1}{3}
\end{aligned}$$

Where:

1. going into (1) we use the fact that $\mathsf{t}^* = \frac{1}{\mathsf{q}^2}$ in expectation, and note we add one extra round as the VA_k is given $\mathfrak{mint}$ as soon as he is selected into the position. And that the minimum wallet balance to enter FeaAudit must be $\mathfrak{v}$.
2. Going into (2), we use set $\mathfrak{v}_\mathsf{k} = \mathfrak{v}$ and note that $-\mathfrak{v}_\mathsf{k}/\mathfrak{v} \leq -1$.

In the expression, we assume the VA_k enters the eunuch position with the minimum required funds $\mathfrak{v}$. In the event where the VA_k's wealth is $\alpha \cdot \mathfrak{v}$ for $\alpha \geq 1$, and the average Witness wealth is $\beta \cdot \mathfrak{v}$, then we should set ξ to:

$$\xi \sim 1 - \frac{2\mathsf{q}^2\alpha}{(1+\mathsf{q}^2)\beta}, \quad \text{where } \alpha, \beta \geq 1. \qquad \text{(C.13)}$$

In a production FeaAudit protocol, it may be advisable to adjust ξ dynamically based on how much funds are in the players' wallet. ■

Proof. for proposition 3.4.1. Given some interval $\mathsf{t}^* \in [1, \mathsf{t}^*_u]$, where t^* denotes the stage where VA_k flips a Head, Head sequence and is thus eliminated. Then we know this corresponds to a binary tree of depth t^*_u, where in every tree-depth $\mathsf{t} \in [1, \mathsf{t}^*_u]$, the left branch terminates on $\mathfrak{p}(\mathsf{Head}, \mathsf{Head}) = \mathsf{q}_\mathsf{t}^2$, while the right branch continues with $\mathfrak{p}(\mathsf{Head}, \mathsf{Tail} \text{ or } \mathsf{Tail}) = 1 - \mathsf{q}_\mathsf{t}^2$. The probability of terminating on any stage t^* is then:

$$\mathfrak{p}(\text{rejected on stage } \mathsf{t}^*) = \mathsf{q}^2_{\mathsf{t}^*}\prod_{\mathsf{t}=1}^{\mathsf{t}^*-1}(1-\mathsf{q}_\mathsf{t}^2).$$

Using eqn C.12 for VA_k's wealth accumulation for terminating on stage t^* modulo the $\mathfrak{v} + \mathsf{t}\Delta\mathfrak{v}$ prefix.

The VA's expected value over the coin tosses with support on interval $[1, \mathsf{t}_u^*]$ is:

$$\begin{aligned}\mathbf{E}\Big[\mathsf{VA_k}\text{'s income on } [1,\mathsf{t}_u^*]\Big] &= \sum_{\mathsf{t}^*=1}^{\mathsf{t}_u^*} \underbrace{\mathsf{q}_{\mathsf{t}^*}^2 \prod_{\mathsf{t}=1}^{\mathsf{t}^*-1}(1-\mathsf{q}_{\mathsf{t}}^2)}_{\mathfrak{p}(\text{rejected on stage } \mathsf{t}^*)} \cdot \underbrace{\frac{\xi}{2}\sum_{\mathsf{t}=0}^{\mathsf{t}^*}\mathfrak{v}_{\mathsf{i}}}_{\text{income if rejected on } \mathsf{t}^*} \\ &\overset{(1)}{\geq} \sum_{\mathsf{t}^*=1}^{\mathsf{t}_u^*} \frac{2^2}{3^2}\left(\frac{5}{9}\right)^{\mathsf{t}^*-1}\frac{\xi}{2}\big(\mathsf{t}^*+1\big)\beta\mathfrak{v} \\ &\overset{(2)}{\geq} 2\beta\mathfrak{v}\xi\cdot\sum_{\mathsf{t}^*=0}^{\mathsf{t}_u^*}\left(\mathsf{t}^*+2\right)\frac{5^{\mathsf{t}^*}}{9^{\mathsf{t}^*+1}}.\end{aligned}$$

Where going into (1), we set the average Witnesses' holding to $\mathfrak{v}_{\mathsf{i}} = \beta\mathfrak{v}$ for $\beta \geq 1$. Then we let $\mathsf{q}_{\mathsf{t}^*} = \frac{2}{3}$ and note that:

$$\mathfrak{p}\left(\text{rejected on stage } \mathsf{t}^* \;\middle|\; \mathsf{q}_{\mathsf{t}^*} = \frac{2}{3}\right) > \mathfrak{p}\left(\text{rejected on stage } \mathsf{t}^* \;\middle|\; \mathsf{q}_{\mathsf{t}^*} = \frac{1}{3}\right).$$

Going into (2), we shift the index so that $\mathsf{t}^* = 0$. Similarly, we can find an upper bound with:

$$\begin{aligned}\mathbf{E}\Big[\mathsf{VA_k}\text{'s income on } [1,\mathsf{t}_u^*]\Big] &\overset{(1)}{\leq} \sum_{\mathsf{t}^*=1}^{\mathsf{t}_u^*}\frac{1^2}{3^2}\left(\frac{8}{9}\right)^{\mathsf{t}^*-1}\frac{\xi}{2}\big(\mathsf{t}^*+1\big)\beta\mathfrak{v} \\ &\leq \frac{\beta\mathfrak{v}\xi}{2}\sum_{\mathsf{t}^*=0}^{\mathsf{t}_u^*}\left(\mathsf{t}^*+2\right)\frac{8^{\mathsf{t}^*}}{9^{\mathsf{t}^*+1}},\end{aligned}$$

where going into (1) we set $\mathsf{q}_{\mathsf{t}^*} = \frac{1}{3}$. ■

APPENDIX D

Sequential Parimutuel Prediction Markets

D.1 Introduction

The heart of QAudit is the prediction market, designed to *elicit truthful report* of information from agents, and compensate agents in accordance to the quality of their report. In this regard, a prediction or information market is similar to a stock market. One common design pattern is to enumerate the outcome set $\Omega = \{.., \omega_i, ..\}$ and offer a security for each outcome ω_i, so that the price reflects the likelihood of the outcome in Ω. For example, in a *permutation betting* prediction market, a designer may elicit the ranking of a three-way horse race between horses $\{1, 2, 3\}$ from the betters. There are $3! = 6$ possible outcomes:

$$\Omega = \{(1,2,3), (1,3,2), (2,1,3), (2,3,1), (3,1,2), (3,2,1)\}.$$

The most expressive market maker would offer one Arrow-Debreu security (binary option) ω_i for each outcome, so that the traders holding the option ω_i receive \$1.00 if the horses finish the order specified by ω_i, and receive nothing otherwise. Each ω_i would be priced by some pricing function score so that the price is reflection of the probability p of the event occurring:

$$\sum_{\omega_i \in \Omega} \mathsf{score}(\omega_i) = 1 \qquad 0 \leq \mathsf{score}(\omega_i) \leq 1. \tag{D.1}$$

A betting market whereby every natural outcome is expressed in a binary option is *complete* [For+07]. Clearly as size of Ω grows, it may be difficult to market every security. Thus it is prudent to express a smaller set of events with $\Omega_a = \{1, 2, 3\}$ stating which horse will finish first, or $\Omega_b = \{1, 2, 3\}$ stating which horse will finish second. The event set Ω_a is called the *winning track*. A pricing function score that elicits the truthful likelihood estimation of events from rational betters is called a *proper scoring rule* [[Han02],[Han03],[ACV12],[GR07],[FPW23]]. Proper scoring rules ensure the agents reporting the information can only profit if they reveal their private truth. That is to say the proper scoring rule is *incentive compatible*: it aligns the agent's interest with those of the market designer. Definition D.1.1 restates this notion in more formal terms.

Definition D.1.1 (Proper Scoring Rule)**.** Let there be a random variable ranging over event space $\Omega = \{\omega\}_n$, let the probability vector $\mathbf{p} = (p_1, .., p_\omega, .., p_n)$ where $\sum_\omega p_\omega = 1$ be a distribution over the set Ω, and let $\mathfrak{P}$ be the set of all possible probability distributions over Ω, ie $\mathbf{p}, \mathbf{r} \in \mathfrak{P}$. A scoring rule:

$$\mathsf{score} : \mathfrak{P} \times \Omega \to \mathbb{R}$$

maps some probability distribution $\mathbf{p} \in \mathfrak{P}$ and event $\omega \in \Omega$ onto the scalar $\mathsf{score}(\mathbf{p}, \omega)$. For example, if a person reports his probability distribution as $\mathbf{r} = (r_1, .., r_\omega, .., r_n)$ and event ω is

realized, then his reward would be $\mathsf{score}(\mathbf{r}, \omega)$. A scoring rule is proper if telling the truth maximizes the score. That is to say score must satisfy the incentive compatibility constraint:

$$\mathbf{p}^* = \arg\max_{\mathbf{r}} \sum_{\omega \in \Omega} \mathsf{p}_\omega \mathsf{score}(\mathbf{r}, \omega), \tag{D.2}$$

and a rational participation constraint:

$$\sum_{\omega} \mathsf{p}_\omega \mathsf{score}(\mathbf{p}, \omega) \geq 0.$$

Moreover, if an expert reports his probability as $\mathbf{r}$, then the scoring rule score is *strictly proper* if reporting $\mathbf{r} = \mathbf{p}$ uniquely maximizes the payout to the expert [GR07].

The next two properties a market designer should satisfy is *budget balance* and *optimal opinion pooling*, incidentally they are mutually reinforcing, and the solution is the *market scoring rule* [Han02]. A market scoring rule is akin to the proper scoring rule, except the payout score is no longer a stateless function, but is instead a function of both the event reported ω_i, and the shared underlying market state $\mathbf{p}$: $\mathsf{score}(\mathbf{p}, \omega_\mathsf{i})$. This underlying market state $\mathbf{p}$ is updated each time an agent buys from the market, and represents the market consensus for a distribution over event space Ω. A sequential parimutuel market works as follows:

1. The agents approach the market sequentially, and report their hidden estimation of likelihood by buying Arrow-Debreu options.
2. The Amm keeps a shared likelihood estimate across reports, reflecting the latest market consensus on the likelihood of an event. The market scoring rule incorporates the Agent's report and updates the likelihood. See table D.1.

Sequential parimutuel market: Agent report $\mathbf{p}$

Stage t	Agent wagers $\mathfrak{v}^\mathsf{t}_{\mathsf{k,i}}$	Agent reports $\mathbf{p}$	reward $\mathfrak{v}^\mathsf{t}_{\mathsf{Agent}}$
0	0	$\mathsf{p}^0_{\mathsf{Amm}} \to \log 0.5$	0
1	$\mathfrak{v}^1_{\mathsf{alice}}$	$\mathsf{p}^1_{\mathsf{alice}} \to \log 0.4$	0
2	$\mathfrak{v}^2_{\mathsf{bob}}$	$\mathsf{p}^2_{\mathsf{bob}} \to \log 0.7$	$\mathfrak{v}^2_{\mathsf{eve}}$
3	$\mathfrak{v}^3_{\mathsf{eve}}$	$\mathsf{p}^3_{\mathsf{eve}} \to \log 0.9$	$\mathfrak{v}_{\mathsf{pa}}$

Table D.1: In sequential prediction market over $\mathsf{t} = 3$ rounds, the market keeps a persistent state $\mathbf{p}$ denoting the likelihood an event will occur (i.e. this event may be whether it will rain on Sunday). Each $\mathsf{Agent}_\mathsf{k}$ approaches the Amm and reports his subjective probability $\mathbf{p}^\mathsf{t}_\mathsf{k}$. The Agents deposit some wager w^t_k as they report to signal their degree of confidence. The Amm then updates the market likelihood to reflect the latest consensus. At test time, nature reveals that it does rain on Sunday. Thus every $\mathsf{Agent}_\mathsf{k}$ who improved the market's estimate is rewarded with the next Agent_s's wager, denoted in green. Agents who degraded the market's estimate lose their wager. If the last Agent reporting improves the market's estimate, then she is rewarded by the bounty $\mathfrak{v}_{\mathsf{pa}}$. Thus the market achieves budget balance, and the most the Amm has to pay is the last Agent to report.

A sequential parimutuel prediction market with shared market scoring rule satisfies the two properties:

1. (*optimal opinion pooling)* If the agent holds a private belief for event ω_i, and judge the market price/likelihood of ω_i's as low, then he will buy the under-priced option,

thus elevating its price. Many agents may buy option ω_{i} until its likelihood under the market scoring rule score matches their subjective beliefs. As the likelihood or price of ω_{i} increases, all other events in Ω will see decrease in likelihood because of constraint D.1 [CV10]. Market scoring rules can pool opinions in both thin markets and thick markets.

a) In the thin market, trading volume is low and the agents may not buy every option ω_{i} on offer, thus rendering the likelihood of some events under-estimated by thin market activity. Market scoring rule ensures that when the likelihood of one event rises, the likelihood of other events fall. Thus the buy-signal propagates to all events.
b) In the thick market, many agents are buying options. The market scoring rule collates their buy signal using the same principle, thus constructing a single source of truth for what the "market believe will occur."

Thus optimal opinion pooling is achieved in all market conditions. The most well-known example of market scoring rule is the *logarithmic market scoring rule* (LMSR), which is used in QAudit [Han02].

2. (*budget balance)* This implies the agents are mostly taking winnings from each other, and the market designer does not have to pay every agent's profit. This is possible because each agent is paying for the previous agent's score. This is the *sequential parimutuel* aspect of the prediction market. Assuming the final observed outcome is ω^*, and the agents report distributions $\{\mathbf{p}_o, \ldots, \mathbf{p}_T\}$ over t rounds, then the payment and payout process can be written algebraically:

$$\begin{aligned}\text{market payout} &= \mathsf{score}(\mathbf{p}_{\mathsf{T}}, \omega^*) - \mathsf{score}(\mathbf{p}_{\mathsf{T}-1}, \omega^*) + \mathsf{score}(\mathbf{p}_{\mathsf{T}-1}, \omega^*) - \ldots - \mathsf{score}(\mathbf{p}_{\mathsf{o}}, \omega^*) \\ &= \mathsf{score}(\mathbf{p}_{\mathsf{T}}, \omega^*) - \mathsf{score}(\mathbf{p}_{\mathsf{o}}, \omega^*).\end{aligned}$$

Thus the market designer is only responsible for paying the difference between the initial report $\mathbf{p}_o$, and the final report $\mathbf{p}_{\mathsf{T}}$. All other agents are paying for each other, this is "budget balance." Overall, the market designer's worst case loss is:

$$\mathsf{max\ payout} = \max_{\omega^* \in \Omega} \max_{\mathbf{p}_{\mathsf{T}} \in \mathfrak{P}} \Big(\mathsf{score}(\mathbf{p}_{\mathsf{T}}, \omega^*) - \mathsf{score}(\mathbf{p}_o, \omega^*) \Big). \tag{D.3}$$

Normally, the market designer would be responsible for paying the value of D.3, thus the market designer is also a *market maker*. However QAudit is a decentralized auditing protocol, and Metheus merely runs the software. Instead any principle auditor launching an instance of QAudit must do so with fiat-denominated bounty computed according to D.3. Thus the PA is the market maker.

In the example given thus far, we assume the market's underlying state is the set of events $\Omega = \{\omega_1, \ldots\}$ and a full distribution $\mathbf{p}$ over events Ω. The market charges the agent to update the shared global market state $\mathbf{p}$. There are many other formats of information that can be elicited from agents, amongst which are distribution parameters and raw data.

1. In machine learning the distribution $\mathfrak{p}$ is typically parameterized by some $\mathbf{w}$. We observe r.v. drawn according to distribution $\mathfrak{p}$ with $\mathbf{x} \in \mathfrak{p}$, and learn a discriminative model with $\mathfrak{p}(\mathsf{y}|\mathbf{x}; \mathbf{w})$, denoting the likelihood that the data $\mathbf{x}$ has label y under parameter $\mathbf{w}$. As long as the model $\mathbf{w}$ is parameterizing a distribution in the exponential family (see Appendix F) with convex log-partition function, then it is possible to build a market scoring rule that is incentive compatible [[ACV12], [AK+14],[CV10]]. The reason is that such models are typically trained by maximizing the log-likelihood of data, which is the same as minimizing the Kullback-Leibler divergence between current

model parameter and the optimal parameter (see remark E.3.4). Thus the log-partition function is a proxy for the cost function. In fact, the logarithmic market scoring rule reduces to the Bregman divergence (or the Kullback-Leibler divergence in primal form [WJ08]) between the two distributions on this log-partition function [AK+14]. See table D.2 below.

Agent reports classifier $\mathbf{w}$

Wager $\mathfrak{v}_{\mathsf{t}}^{\mathsf{k}}$	Data transformation	Amm shared state
$\mathfrak{v}_1^{\mathsf{alice}}$	$\mathbf{w}_{\mathsf{alice}}^1 \to \log \mathbf{p}_{\mathsf{alice}}^1$	latest reported $\mathbf{w}$

Table D.2: If the Agent reports some $\mathbf{w}^{\mathsf{t}}$ at round t parameterizing a logistic regression model, then it is possible to compute a likelihood $\mathbf{p} = \mathfrak{p}(\mathsf{y} = 1|\mathbf{x}; \mathbf{w}^{\mathsf{t}}) = \boldsymbol{\sigma}(\mathbf{w}^{\intercal}\mathbf{x})$ given data point $(\mathbf{x}, y)$. If the model is drawn from the exponential family, then its computed probability $\mathbf{p}$ is s.t budget balance can be achieved in expectation. The Amm's shared state is the latest $\mathbf{w}^{\mathsf{t}}$.

2. In the event where the elicited model $\boldsymbol{\theta}$ does not parameterize an exponential family distribution, as is the case when $\mathfrak{p}(\mathbf{x}; \boldsymbol{\theta})$ is a deep artificial neural network (ANN), then we must score a convex surrogate instead. Often times, the most commercially interesting ANNs such as large language models (LLMs) and image generation models do not output a probability at all, instead they output real values conditioned on inputs. LLM for example is of form $\mathfrak{p}(\mathbf{x}_t|\mathbf{x}_{\mathsf{t}-1}; \theta)$, that is to say the model is parameterized by $\boldsymbol{\theta}$, and receive prompts $\mathbf{x}_{\mathsf{t}-1}$ and generate response $\mathbf{x}_{\mathsf{t}}$. Thus the surrogacy must score the likelihood of response given query. The surrogacy may be a logistic regression model that determines the quality of the generated response.

Agent reports generative model $\boldsymbol{\theta}$

Wager $\mathfrak{v}_{\mathsf{t}}^{\mathsf{k}}$	Data transformation	Amm shared state
$\mathfrak{v}_1^{\mathsf{alice}}$	$\boldsymbol{\theta}_{\mathsf{alice}}^1 \to \mathbf{w}_{\mathsf{alice}}^1 \to \log \mathbf{p}_{\mathsf{alice}}^1$	$\mathbf{w}$ trained on $\boldsymbol{\theta}$

Table D.3: When Agents report some generative model $\boldsymbol{\theta}$ or raw data $\mathbb{D}$, then QAudit samples from the submission and build a classifier $\mathbf{w}$, and scores the quality of this classifier at discriminating "good" vs. "bad" data. The Amm's shared state is the $\mathbf{w}^{\mathsf{t}}$ trained from the latest submitted $\boldsymbol{\theta}^t$.

3. Agents can also report raw data. Similar to the previous point, raw data does not have some cost function attached, but it can be used to train a convex surrogate for scoring. A QAudit that is eliciting data from agents is in fact running a data market and buying information from users. This is an important aspect of the Metheus economy, and significantly expand the surface area of who can participate in the games. That is most people cannot train machine learning models, but most people do have access to some data.

4. Finally, Agents may not report a distribution at all, but rather buy shares of some underlying event. This is the most natural case for Agents who perceive the prediction market as a stock market. In this case the Amm keeps an underlying state that is the total shares owed to all Agent, denoted with $\mathbf{s}$. This market inventory can be turned into a proper distribution $\mathbf{p}$ with the LMSR in example D.1.2.

Agent reports buy vector $\mathbf{b}$

Wager $\mathfrak{v}_t^k$	Data transformation	Amm shared state
$\mathfrak{v}_1^{\mathsf{alice}}$	$\mathbf{b}^1_{\mathsf{alice}} \to \mathbf{p}^1_{\mathsf{alice}}$	$\mathbf{s} = (\mathsf{s}_1^t, \ldots, \mathsf{s}_K^t)$

Table D.4: When Agents report buy vector for shares, the market consensus likelihood is updated by logarithmic market scoring rule.

Finally, we draw a connection between sequential prediction markets and online learning. Since the prediction market is collating predictions from experts sequentially, it is in effect an online learning algorithm leveraging a mixture of experts to arrive at some final prediction [[ACV12], [Rak09]]. Thus the QAudit auditing process is fundamentally oriented around monetizing online learning from mixture of experts, where every Agent is an expert. See [Rak09] and [CL06] for further discussion.

Example D.1.2. (*Logarithmic market scoring rule: LMSR*) QAudit use LMSR as a pricing function [[CA10],[ACV12],[CP07]]. Defined with:

$$\rho_\omega = \mathsf{score}(\omega; \mathbf{s}) = \frac{\exp\big(\mathsf{s}_\omega/b\big)}{\sum_{\nu\in\Omega}\exp\big(\mathsf{s}_\nu/b\big)}.$$

And observe $0 \leq \rho_\omega \leq 1$. Now let $b = 1$ and $\Omega = \{a, b\}$, and setup an automated market maker (Amm) that issues Arrow-Debreu securities so that the market pays out \$1 if ω occurs. Then initialize the Amm with state:

$$\mathbf{s}^{\mathsf{o}} = (10, 10).$$

The initial market cost is $\mathsf{cost}(\mathbf{s}^{\mathsf{o}}) = \log(e^{10} + e^{10}) = \$4.64$. And its price per option is:

$$\rho_{\mathsf{a}} = \mathsf{score}(\mathsf{a}; \mathbf{s}^{\mathsf{o}}) = \frac{e^{10}}{e^{10}+e^{10}} = \$0.50 \qquad \rho_{\mathsf{b}} = \mathsf{score}(\mathsf{b}; \mathbf{s}^{\mathsf{o}}) = \frac{e^{10}}{e^{10}+e^{10}} = \$0.50.$$

Now suppose Alice places buys 3 shares of a and one share of b, so that is her order is $\mathbf{b}_{\mathsf{alice}} = (3, 1)$. According to the posted price, she is charged:

$$\Big\langle \mathbf{b}_{\mathsf{alice}},\ \nabla\mathsf{cost}(\mathbf{s}^{\mathsf{o}}) \Big\rangle = (3,1)^{\intercal}(0.50, 0.50) = \$2.00.$$

Now the market state is:

$$\mathbf{s}^1 = \mathbf{s}^{\mathsf{o}} + \mathbf{b}_{\mathsf{alice}} = (13, 11),$$

where we use element wise addition. And the prices at $\mathbf{s}^1$ becomes:

$$\rho_{\mathsf{a}} = \mathsf{score}(\mathsf{a}; \mathbf{s}^1) = \frac{e^{13}}{e^{13}+e^{11}} = \$0.881 \qquad \rho_{\mathsf{b}} = \mathsf{score}(\mathsf{b}; \mathbf{s}^1) = \frac{e^{11}}{e^{13}+e^{11}} = \$0.119.$$

And note how purchasing more shares of a increases the price of a and decreases the price of b. Finally let $\omega^* = a$, then the payout outcome vector is $(1, 0)$, and Alice's payout is:

$$\Big\langle (1,0),\ \mathbf{b}_{\mathsf{alice}} \Big\rangle = (1,0)^{\intercal}(3,1) = \$3.00.$$

Her profit is $\$3.00 - \$2.00 = \$1.00$. In the event where the payout vector is $(0, 1)$, her payout is:

$$\Big\langle (0,1),\ \mathbf{b}_{\mathsf{alice}} \Big\rangle = (0,1)^{\intercal}(3,1) = \$1.00.$$

So her profit is $\$1.00 - \$2.00 = -\$1.00$. Next suppose bob posts buy order $\mathbf{b}_{bob} = (-2, 5)$, stating he shorts two shares of a and longs 5 shares of b, then he is charged:

$$(-2, 5)^\intercal (0.881, 0.119) = -\$1.166,$$

so in this case the market pays Bob for his order. The market state becomes $\mathbf{s}^2 = (11, 16)$ and updates prices to:

$$\mathsf{p}(\mathsf{a}; \mathbf{s}^2) = \frac{e^{11}}{e^{16} + e^{11}} = \$0.007 \qquad \mathsf{p}(\mathsf{b}; \mathbf{s}^2) = \frac{e^{16}}{e^{16} + e^{11}} = \$0.993.$$

Finally suppose $\omega^* = a$ so that the payout vector is $(1, 0)$, then Bob's payout is $(1, 0)^\intercal (-2, 1) = -\$2.00$. Bob's loss on this trade is $\$1.166 - \$2.00 = -\$0.834$, and he has to pay from his balance cover this bet. If $\omega^* = b$, then Bobs' payout is $(0, 1)^\intercal \cdot (-2, 1) = \$1.00$, and he gains \$2.166 on this trade. If Bob is the last agent to trade with the protocol, then the Amm has to pay Bob the \$1.00, this is Bob's mining reward.

D.2 Exponential Family Sequential Prediction Market

In this case, the distribution cannot be used in downstream applications. Instead the agents share *one* underlying model $\boldsymbol{\theta}$ that parameterizes the distribution generating the data. In lieu of issuing Arrow-Debreu options for every event $\omega \in \Omega$, the Amm sells permissions to update the shared model.

Algorithm 15 Exponential Family Sequential Prediction Market

Require: bounty $\mathfrak{v}_{\mathsf{pa}} \geq 0$, instance of the exponential family with $(\boldsymbol{\varphi}, A)$
Require: private dataset $\mathbb{D} = \mathbb{D}_{\mathsf{train}} \cup \mathbb{D}_{\mathsf{test}} = \{\mathbf{x}_{train,\mathsf{i}}\} \cup \{\mathbf{x}_{test,\mathsf{i}}\}$
Require: reveals training set $\mathbb{D}_{\mathsf{train}} = \{\mathbf{x}_{train,\mathsf{i}}\}$
Require: each Agent k has balance a_{k} so that
  $a_{\mathsf{k}} \leftarrow 0$ for $\mathsf{k} \in [1...\mathsf{K}]$
  $\boldsymbol{\theta}_0 \leftarrow \tau \mathbf{1}$
  $\Theta_0 \leftarrow [\boldsymbol{\theta}_0\,]$
  $b_{\mathsf{Amm}} \leftarrow \mathfrak{v}_{\mathsf{pa}}$
  for $\mathsf{k} \in [1...\mathsf{K}]$ **do**
    $\mathsf{Agent}_{\mathsf{k}}$ query Amm state and training set: $(\boldsymbol{\theta}_{\mathsf{k}-1}, \{\mathbf{x}_{train,\mathsf{i}}\})$
    $\mathsf{Agent}_{\mathsf{k}}$ submit updated model $\boldsymbol{\theta}_{\mathsf{k}}$ to market and receive price:
      $\rho_{\mathsf{k}} = \mathbf{mc}(\rho(\cdot, \boldsymbol{\theta}_{\mathsf{k}}, \boldsymbol{\theta}_{\mathsf{k}-1}; \boldsymbol{\varphi}, A), \{\mathbf{x}_{train,\mathsf{i}}\})$
    $\mathsf{Agent}_{\mathsf{k}}$ is charged ρ_{k}
    Amm updates state with:
      $\boldsymbol{\theta}_{\mathsf{k}-1} \leftarrow \boldsymbol{\theta}_{\mathsf{k}}$
      $\Theta_{\mathsf{k}-1} \leftarrow \Theta_{\mathsf{k}-1} \cup [\boldsymbol{\theta}_{\mathsf{k}}]$
      $b_{\mathsf{Amm}} \leftarrow b_{\mathsf{Amm}} + \rho_{\mathsf{k}}$
  end for
  Reveals test set $\mathbb{D}_{\mathsf{test}}$
  for $\mathsf{k} \in [1...\mathsf{K}]$ **do**
    $a_{\mathsf{k}} \leftarrow \sum_{\mathbf{x}_{test,\mathsf{i}} \in \mathbb{D}_{\mathsf{test}}} \mathsf{score}(\mathbf{x}_{test,\mathsf{i}}, \boldsymbol{\theta}_{\mathsf{k}}, \boldsymbol{\theta}_{\mathsf{k}-1}; \boldsymbol{\varphi}, A)$
  end for

For example the data may be generated by a Bernoulli distribution over $\{0, 1\}$, in which case the market state is the likelihood the proverbial coin will land head; or the data may be generated by a Gaussian distribution, whereby the parameters are sample mean and variance. Following the convention of [AK+14], we call this type of market the *exponential family prediction market*. The exponential family is of form:

$$\mathsf{p}(\mathbf{x}; \boldsymbol{\theta}) = \exp\big(\langle \boldsymbol{\varphi}(\mathbf{x}), \boldsymbol{\theta} \rangle - \mathsf{A}(\boldsymbol{\theta})\big) \text{ where } \mathsf{A}(\boldsymbol{\theta}) = \log \int_{\Omega^n} \exp\big(\langle \boldsymbol{\varphi}(\mathbf{x}), \boldsymbol{\theta} \rangle\big) \nu d(\mathbf{x}).$$

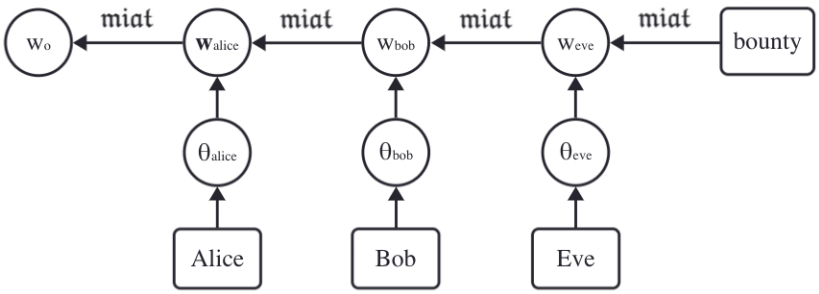


Figure D.1: In the exponential family sequential prediction market, the protocol keeps a common state $\mathbf{w}$, which parametrize a discriminative model $\mathfrak{p}(y|\mathbf{x};\mathbf{w})$. Agents such as alice the approaches the market and submits their generative model $\boldsymbol{\theta}_{\mathsf{alice}}$. The protocol then trains a discriminative model $\mathbf{w}_{\mathsf{alice}}$ from this $\boldsymbol{\theta}_{\mathsf{alice}}$, and updates the protocol state $\mathbf{w}$.

Where $\boldsymbol{\varphi}$ is the sufficient statistic associated with data sampled from distribution over Ω^n, A is the log partition function, and $\boldsymbol{\theta}$ is the *canonical parameter* of the distribution. The incentives in this market are of flavor:

$$\rho(\mathbf{x}, \boldsymbol{\theta}_{\mathsf{k}}, \boldsymbol{\theta}_{\mathsf{k}-1}; \boldsymbol{\varphi}, A) = \mathbf{E}_{\mathfrak{p}(\cdot;\boldsymbol{\theta}_{\mathsf{k}-1})}\left[\mathsf{score}(\mathbf{x}, \boldsymbol{\theta}_{\mathsf{k}}, \boldsymbol{\theta}_{\mathsf{k}-1}; \boldsymbol{\varphi}, A)\right] = -\mathsf{D}(\boldsymbol{\theta}_{\mathsf{k}-1}||\boldsymbol{\theta}_{\mathsf{k}}) \tag{D.4}$$

$$\mathsf{score}(\mathbf{x}, \boldsymbol{\theta}_{\mathsf{k}}, \boldsymbol{\theta}_{\mathsf{k}-1}; \boldsymbol{\varphi}, A) = \boldsymbol{\varphi}(\mathbf{x})^{\intercal}(\boldsymbol{\theta}_{\mathsf{k}} - \boldsymbol{\theta}_{\mathsf{k}-1}) + \mathsf{A}(\boldsymbol{\theta}_{\mathsf{k}-1}) - \mathsf{A}(\boldsymbol{\theta}_{\mathsf{k}}) \tag{D.5}$$

Where $\mathbf{x}$ is the data point sampled according to distribution $\mathfrak{p}$. The vector $\boldsymbol{\theta}_{\mathsf{k}}$ is the kth submission of model parameters. $\mathsf{D}(\cdot||\cdot)$ is the Bregman divergence between two distributions parameterized by $\boldsymbol{\theta}_{\mathsf{k}}$ and $\boldsymbol{\theta}_{\mathsf{k}-1}$, and A is the log partition function. Note these are class level definitions using canonical parameterization of the exponential family. It is also possible to rewrite the price and rewards in the mean forms, though we omit them until needed.

The expression $\mathbf{mc}(\mathsf{score}(\cdot), \{\mathbf{x}_{train,\mathsf{i}}\})$ refers to some sampling function using examples $\mathbf{x}_{train,\mathsf{i}}$ to estimate the KL divergence (read: score). If the KL divergence has closed form expression where mean parameters are sufficient, then $\mathbf{mc}$ is the identity function:

$$\mathbf{mc}(\mathsf{score}(\cdot), \{\mathbf{x}_{train,\mathsf{i}}\}) = \mathsf{score}(\cdot).$$

The Exponential Family Market Maker: Part I

A basic understanding of exponential family distribution is needed to understand this market. The best source is [WJ08], there is brief introduction in Appendix F that situates the topic within prediction markets. In particular, the pricing mechanism in a sequential prediction market involves comparing the quality of the current prediction against the previous one, in the context of the final outcome at market closing. Thus the motivating question here is how to price the difference in quality between two models of the exponential family, and how various representations of exponential family can be used in said pricing. This section derives the ρ and score function from LMSR. But instead of using the prior formulation where the entire distribution $\mathbf{p}$ is reported, [AK+14] showed it is possible to arrive at a proper scoring rule, with a incentive compatible measure of agent reports - without recording the full distribution as market state.

Recall the log scoring rule parameterized by the full distribution is $\mathsf{score}(\mathbf{p}, \omega) = \log(\mathbf{p}_{\omega})$. Now instead of eliciting the full distribution $\mathbf{p}$, only the canonical parameter $\boldsymbol{\theta}$ is reported. Then the scoring rule can be rewritten as:

$$\mathsf{score}(\boldsymbol{\theta}, \mathbf{x}) = \log(\mathfrak{p}(\mathbf{x}; \boldsymbol{\theta})),$$

for some realization of the vector $\mathbf{x} \in \Omega^n$, drawn according to $\mathfrak{p} \in \mathfrak{P}$ with parameter $\boldsymbol{\theta}$. This form is only usable if the canonical representation satisfies the same property as eliciting the full distribution. In particular, the canonical form must satisfy the incentive capability condition: given the agent's truthful $\boldsymbol{\theta}_1$, and some $\boldsymbol{\theta}_2 \neq \boldsymbol{\theta}_1$, with $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2 \in \Theta$, score is incentive compatible if:

$$\mathbf{E}_{\mathfrak{p}(\cdot;\boldsymbol{\theta}_1)}[\log(\mathfrak{p}(\mathbf{x};\boldsymbol{\theta}_1))] - \mathbf{E}_{\mathfrak{p}(\cdot;\boldsymbol{\theta}_1)}[\log(\mathfrak{p}(\mathbf{x};\boldsymbol{\theta}_2))] \geq 0. \tag{D.6}$$

It turns out, this is true for LMSR only if $\mathfrak{p}$ is in the exponential family.

Theorem D.2.1. *Take the log scoring rule* $\mathsf{score}(\boldsymbol{\theta}, \mathbf{x}) = \log(\mathfrak{p}(\mathbf{x};\boldsymbol{\theta}))$ *defined on densities parameterized by* $\boldsymbol{\theta} \in \Theta$*, it is proper if the densities are in the exponential family.*

Proof. Going in the forward direction, assume $\mathfrak{p}$ is in the exponential family, then $\log(\mathfrak{p}(\mathbf{x};\boldsymbol{\theta})) = \boldsymbol{\varphi}(\mathbf{x})^\intercal \boldsymbol{\theta} - \mathsf{A}(\boldsymbol{\theta})$. And we show the expected value of reporting the truth is no lesser than the expected value of reporting a lie. Let $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2 \in \Theta$ be the agent's true belief and lie respectively, then:

$$\begin{aligned}
&\mathbf{E}_{\mathfrak{p}(\cdot;\boldsymbol{\theta}_1)}[\log(\mathfrak{p}(\mathbf{x};\boldsymbol{\theta}_1))] - \mathbf{E}_{\mathfrak{p}(\cdot;\boldsymbol{\theta}_1)}[\log(\mathfrak{p}(\mathbf{x};\boldsymbol{\theta}_2))] \\
&= \mathbf{E}_{\mathfrak{p}(\cdot;\boldsymbol{\theta}_1)}\Big[\boldsymbol{\varphi}(\mathbf{x})^\intercal \boldsymbol{\theta}_1 - \mathsf{A}(\boldsymbol{\theta}_1) - \big(\boldsymbol{\varphi}(\mathbf{x})^\intercal \boldsymbol{\theta}_2 - \mathsf{A}(\boldsymbol{\theta}_2)\big)\Big] \\
&= \int_{\Omega^n} \mathfrak{p}(\mathbf{x};\boldsymbol{\theta}_1)\Big(\mathsf{A}(\boldsymbol{\theta}_2) - \mathsf{A}(\boldsymbol{\theta}_1) + (\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2)^\intercal \boldsymbol{\varphi}(\mathbf{x})\Big)\nu(d\mathbf{x}) \\
&= \mathsf{A}(\boldsymbol{\theta}_2) - \mathsf{A}(\boldsymbol{\theta}_1) + (\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2)^\intercal \int_{\Omega^n} \mathfrak{p}(\mathbf{x};\boldsymbol{\theta}_1)\boldsymbol{\varphi}(\mathbf{x})\nu(d\mathbf{x}) \\
&= \mathsf{A}(\boldsymbol{\theta}_2) - \mathsf{A}(\boldsymbol{\theta}_1) - \nabla\mathsf{A}(\boldsymbol{\theta}_1)^\intercal(\boldsymbol{\theta}_2 - \boldsymbol{\theta}_1).
\end{aligned} \tag{D.7}$$

Where going into the last line we use the moment matching condition to set:

$$\mathbf{E}_{\mathfrak{p}(\cdot;\boldsymbol{\theta}_1)}[\boldsymbol{\varphi}(\mathbf{x})] = \int_{\Omega^n} \mathfrak{p}(\mathbf{x};\boldsymbol{\theta}_1)\boldsymbol{\varphi}(\mathbf{x})\nu(d\mathbf{x}) = \boldsymbol{\mu}_1 = \nabla\mathsf{A}(\boldsymbol{\theta}_1).$$

Now note because $\mathfrak{p}$ is in the exponential family with convex $\mathbb{A}$, the value of eqn (D.7) is ≥ 0 by definition, thus proving incentive compatibility condition. Furthermore, the gap between telling the truth and telling a lie is the Bregman divergence between the two $\boldsymbol{\theta}$'s on $\mathsf{A}(\boldsymbol{\theta})$. Now going in the opposite direction, assume the scoring rule is proper, then it must be convex. That is to say for some strictly convex function G, Bregman divergence must hold:

$$\mathsf{score}(\mathbf{p}, \omega) = G(\mathbf{p}) - \nabla G(\mathbf{p})^\intercal(\mathbf{p} - f(\omega)).$$

Setting $\mathbf{p} = \boldsymbol{\mu}$, $\omega = \mathbf{x}$, $G = \mathsf{A}^*$, and $f = \boldsymbol{\varphi}$, we use the dual form of the scoring rule $\log(\mathfrak{p}(\mathbf{x}, \boldsymbol{\mu}))$:

$$\begin{aligned}
\log(\mathfrak{p}(\mathbf{x}, \boldsymbol{\mu})) &= \mathsf{A}^*(\boldsymbol{\mu}) - \nabla\mathsf{A}^*(\boldsymbol{\mu})^\intercal(\boldsymbol{\mu} - \boldsymbol{\varphi}(\mathbf{x})) \\
&= \boldsymbol{\mu}^\intercal\boldsymbol{\theta}_\mu - \mathsf{A}(\boldsymbol{\theta}_\mu) - \boldsymbol{\theta}_\mu^\intercal(\boldsymbol{\mu} - \boldsymbol{\varphi}(\mathbf{x})) \\
&= \boldsymbol{\varphi}(\mathbf{x})^\intercal\boldsymbol{\theta}_\mu - \mathsf{A}(\boldsymbol{\theta}_\mu).
\end{aligned}$$

Where going into the second line, we the use conjugate definition of $A^*(\boldsymbol{\mu})$, and the fact that $\nabla\mathsf{A}^*(\boldsymbol{\mu}) = \boldsymbol{\theta}_\mu$ by condition (3) of the properties of A^* (see Appendix F [AK+14].) ■

Informally, there is a correspondence between incentive compatibility of LMSR, and maximum likelihood estimation in models based on exponential family distributions:

1. every proper scoring rule can be described with an equivalent convex function;
2. the exponential family's log partition function $\mathsf{A}(\boldsymbol{\theta})$ is convex;

3. the gradient of the log partition function is the distribution's expected value;
4. during model training, maximizing the likelihood of data under model $\boldsymbol{\theta}$ is equivalent to minimizing the KL-divergence of the true $\boldsymbol{\theta}^*$ and model $\boldsymbol{\theta}$ on the $\mathsf{A}(\boldsymbol{\theta})$ surface;
5. in LMSR, the agent maximizes his expected payoff by minimizing the KL divergence between his reported value and private value.

Now we can derive the reward score and price ρ functions for each $\mathbf{x}$:

$$\begin{aligned}\mathsf{score}(\mathbf{x}, \boldsymbol{\theta}_\mathsf{k}, \boldsymbol{\theta}_{\mathsf{k}-1}; \boldsymbol{\varphi}, A) &= \log(\mathfrak{p}(\mathbf{x}; \boldsymbol{\theta}_\mathsf{k})) - \log(\mathfrak{p}(\mathbf{x}; \boldsymbol{\theta}_{\mathsf{k}-1})) \\ &= \Big(\boldsymbol{\varphi}(\mathbf{x})^\intercal \boldsymbol{\theta}_\mathsf{k} - \mathsf{A}(\boldsymbol{\theta}_\mathsf{k})\Big) - \Big(\boldsymbol{\varphi}(\mathbf{x})^\intercal \boldsymbol{\theta}_{\mathsf{k}-1} - \mathsf{A}(\boldsymbol{\theta}_{\mathsf{k}-1})\Big) \\ &= \boldsymbol{\varphi}(\mathbf{x})^\intercal(\boldsymbol{\theta}_\mathsf{k} - \boldsymbol{\theta}_{\mathsf{k}-1}) + \mathsf{A}(\boldsymbol{\theta}_{\mathsf{k}-1}) - \mathsf{A}(\boldsymbol{\theta}_\mathsf{k}).\end{aligned}$$

Similarly, ρ is the expected value of the reward under the market's current $\mathfrak{p}$:

$$\begin{aligned}\mathfrak{p}(\mathbf{x}, \boldsymbol{\theta}_\mathsf{k}, \boldsymbol{\theta}_{\mathsf{k}-1}; \boldsymbol{\varphi}, A) &= \mathbf{E}_{\mathfrak{p}(\cdot;\boldsymbol{\theta}_{\mathsf{k}-1})}\Big[\log(\mathfrak{p}(\mathbf{x}; \boldsymbol{\theta}_\mathsf{k})) - \log(\mathfrak{p}(\mathbf{x}; \boldsymbol{\theta}_{\mathsf{k}-1}))\Big] \\ &= \int_{\Omega^n} \mathfrak{p}(\mathbf{x}; \boldsymbol{\theta}_{\mathsf{k}-1})\Big(\boldsymbol{\varphi}(\mathbf{x})^\intercal(\boldsymbol{\theta}_\mathsf{k} - \boldsymbol{\theta}_{\mathsf{k}-1}) + \mathsf{A}(\boldsymbol{\theta}_{\mathsf{k}-1}) - \mathsf{A}(\boldsymbol{\theta}_\mathsf{k})\Big)\nu(d\mathbf{x}) \\ &= \mathsf{A}(\boldsymbol{\theta}_{\mathsf{k}-1}) - \mathsf{A}(\boldsymbol{\theta}_\mathsf{k}) + (\boldsymbol{\theta}_\mathsf{k} - \boldsymbol{\theta}_{\mathsf{k}-1})^\intercal \int_{\Omega^n} \mathfrak{p}(\mathbf{x}; \boldsymbol{\theta}_{\mathsf{k}-1})\boldsymbol{\varphi}(\mathbf{x})\nu(d\mathbf{x}) \\ &= -\Big(\mathsf{A}(\boldsymbol{\theta}_\mathsf{k}) - \mathsf{A}(\boldsymbol{\theta}_{\mathsf{k}-1}) - \nabla\mathsf{A}(\boldsymbol{\theta}_{\mathsf{k}-1})^\intercal(\boldsymbol{\theta}_\mathsf{k} - \boldsymbol{\theta}_{\mathsf{k}-1})\Big) \\ &= -\mathsf{D}(\boldsymbol{\theta}_{\mathsf{k}-1}||\boldsymbol{\theta}_\mathsf{k}).\end{aligned}$$

Where going into the second to last line, we use the moment matching condition of exponential family to set

$$\int_{\Omega^n} \mathfrak{p}(\mathbf{x}; \boldsymbol{\theta}_{\mathsf{k}-1})\boldsymbol{\varphi}(\mathbf{x})\nu(d\mathbf{x}) = \mathbf{E}_{\mathfrak{p}(\cdot;\boldsymbol{\theta}_{\mathsf{k}-1})}[\boldsymbol{\varphi}(\mathbf{x})] = \nabla\mathsf{A}(\boldsymbol{\theta}_{\mathsf{k}-1}).$$

The function ρ is the Bregman divergence between two distribution parameterized by $\boldsymbol{\theta}$ (see figure **??**). This makes intuitive sense, $\mathsf{Agent}_\mathsf{k}$ is paying to update the model $\boldsymbol{\theta}$, and similar to previous protocols, the more he changes the distribution the more he pays. Given the score function, the market maker's payout under Protocol 15 is:

$$\begin{aligned}&\sum_{\mathsf{k}=1}^{\mathsf{K}} \boldsymbol{\varphi}(\mathbf{x})^\intercal(\boldsymbol{\theta}_\mathsf{k} - \boldsymbol{\theta}_{\mathsf{k}-1})^\intercal + \mathsf{A}(\boldsymbol{\theta}_{\mathsf{k}-1}) - \mathsf{A}(\boldsymbol{\theta}_\mathsf{k}) \\ &= \Big(\boldsymbol{\varphi}(\mathbf{x})^\intercal(\cancel{\boldsymbol{\theta}_1} - \boldsymbol{\theta}_0) + \mathsf{A}(\boldsymbol{\theta}_0) - \cancel{A(\boldsymbol{\theta}_1)}\Big) + \Big(\boldsymbol{\varphi}(\mathbf{x})^\intercal(\cancel{\boldsymbol{\theta}_2} - \cancel{\boldsymbol{\theta}_1}) + \mathsf{A}(\cancel{\boldsymbol{\theta}_1}) - \cancel{A(\boldsymbol{\theta}_2)}\Big) + ... \\ &= \boldsymbol{\varphi}(\mathbf{x})^\intercal(\boldsymbol{\theta}_\mathsf{K} - \boldsymbol{\theta}_0) + \mathsf{A}(\boldsymbol{\theta}_0) - \mathsf{A}(\boldsymbol{\theta}_\mathsf{K}).\end{aligned} \tag{D.8}$$

That is to say the Amm's loss depend only on the quality of the last model vs the initial model parameter $\boldsymbol{\theta}_0$. Thus the parimutuel nature of LMSR is preserved. Note eqn D.8 is in fact the KL divergence between $\boldsymbol{\theta}_\mathsf{k}$ and $\boldsymbol{\theta}_0$: $\mathsf{D}(\mathfrak{p}(\mathbf{x}; \boldsymbol{\theta}_0)||\mathfrak{p}(\mathbf{x}; \boldsymbol{\theta}_\mathsf{K}))$. The market maker's worst case loss is then:

$$\sup_{\mathbf{x}\in\Omega^n} \sup_{\boldsymbol{\theta}_\mathsf{K}\in\Theta} \Big(\boldsymbol{\varphi}(\mathbf{x})^\intercal(\boldsymbol{\theta}_\mathsf{K} - \boldsymbol{\theta}_0) + \mathsf{A}(\boldsymbol{\theta}_0) - \mathsf{A}(\boldsymbol{\theta}_\mathsf{K})\Big). \tag{D.9}$$

Where $\mathbf{x}$ refers to the worst draw of test sample the Amm could encounter, and $\boldsymbol{\theta}_\mathsf{K} \in \Theta$ refers to the best performing model parameter from $\Theta = \big\{\boldsymbol{\theta} \in \mathbb{R}^d : \mathsf{A}(\theta) < \infty\big\}$. In

production, this would be the maximum mining reward. Note we can decrease this worst case loss by initializing some $\boldsymbol{\theta}_0$ that is already pre-trained. In other words, instances of Protocol 15 launched to improve pre-trained models have lower mining rewards.

Now observe the pricing function poses an odd quandary for the agent: he can minimizes the price ρ for an update by minimizing the divergence between his reported $\boldsymbol{\theta}_{\mathsf{k}}$, and the market $\boldsymbol{\theta}_{\mathsf{k}-1}$. Therefore even if he believes the market state is wrong, he is incentivized to *not* report his true $\boldsymbol{\theta}$ should he deem the write-privilege too expensive. However, reporting a conservative (read: *less risky*) $\boldsymbol{\theta}$ may also lead to lower profit. Specifically, the agent's profit for some test sample $\mathbf{x}_{test}$ is:

$$\underbrace{\Big(\boldsymbol{\varphi}(\mathbf{x}_{test})^{\intercal}(\boldsymbol{\theta}_{\mathsf{k}}-\boldsymbol{\theta}_{\mathsf{k}-1})+\mathsf{A}(\boldsymbol{\theta}_{\mathsf{k}-1})-\mathsf{A}(\boldsymbol{\theta}_{\mathsf{k}})\Big)}_{\text{payout}}-\underbrace{\Big(\mathsf{A}(\boldsymbol{\theta}_{\mathsf{k}})-\mathsf{A}(\boldsymbol{\theta}_{\mathsf{k}-1})-\nabla\mathsf{A}(\boldsymbol{\theta}_{\mathsf{k}-1})^{\intercal}(\boldsymbol{\theta}_{\mathsf{k}}-\boldsymbol{\theta}_{\mathsf{k}-1})\Big)}_{\text{price}}$$
$$=\Big(\boldsymbol{\varphi}(\mathbf{x}_{test})+\nabla\mathsf{A}(\boldsymbol{\theta}_{\mathsf{k}-1})\Big)^{\intercal}(\boldsymbol{\theta}_{\mathsf{k}}-\boldsymbol{\theta}_{\mathsf{k}-1})+\Big(2A(\boldsymbol{\theta}_{\mathsf{k}-1})-2\mathsf{A}(\boldsymbol{\theta}_{\mathsf{k}})\Big)$$

And we see that reporting a $\boldsymbol{\theta}_{\mathsf{k}}\sim\boldsymbol{\theta}_{\mathsf{k}-1}$ also degrades his reward, so there is a trade off here that can be expressed. Assume the agent has some exponential utility function U:

$$U_a(w)=-\frac{1}{a}e^{-aw}.$$

Where w is the agent's wealth, and a is the agent's risk aversion parameters. Observe as a increases the agent's risk aversion increases. And when $a\to 0$, the agent approaches risk neutrality. [AK+14] showed that given a LMSR market with market state $\boldsymbol{\theta}_{\mathsf{k}-1}$, an agent with private belief $\boldsymbol{\theta}^*_{\mathsf{k}}$ and risk aversion a_{k} has optimal state update:

$$\boldsymbol{\theta}_{\mathsf{k}}=\frac{1}{1+a_{\mathsf{k}}}\boldsymbol{\theta}^*_{\mathsf{k}}+\frac{a_{\mathsf{k}}}{1+a_{\mathsf{k}}}\boldsymbol{\theta}_{\mathsf{k}-1}. \tag{D.10}$$

In particular, note that a risk neutral agent with $a_{\mathsf{k}}=0$ reports the truth: $\boldsymbol{\theta}_{\mathsf{k}}=\boldsymbol{\theta}^*_{\mathsf{k}}$. Where as a risk averse agent with large a_{k} will only incrementally update the model state so that $\boldsymbol{\theta}_{\mathsf{k}}\sim\boldsymbol{\theta}_{\mathsf{k}-1}$. Now suppose each agent k has risk aversion parameter a_{k}, private truth $\boldsymbol{\theta}^*_{\mathsf{k}}$, and updates the model by $\tilde{\boldsymbol{\delta}}_{\mathsf{k}}$, then after K updates the final market state will be:

$$\boldsymbol{\theta}_{\mathsf{K}}=\boldsymbol{\theta}_0+\sum_{\mathsf{k}=1}^{\mathsf{K}}\tilde{\boldsymbol{\delta}}_{\mathsf{k}}=\frac{\boldsymbol{\theta}_0+\sum_{\mathsf{k}=1}^{\mathsf{K}}\frac{\boldsymbol{\theta}^*_{\mathsf{k}}}{a_{\mathsf{k}}}}{1+\sum_{\mathsf{k}=1}^{\mathsf{K}}\frac{1}{a_{\mathsf{i}}}}.$$

So that the market equilibrium state is a convex combination of initial market state, and all the agent's beliefs weighed by their risk aversion [AK+14].

Exponential Family Market Maker: Part II

The previous section characterized log scoring rule using the exponential family, and showed that the LMSR properties are preserved. The canonical form reveals clear expression of incentive compatibility, but the mean parameterization is more natural in computational settings. This section explores LMSR's dual cost function representation. Specifically, the relationship between the cost function cost for generalized LMSR, and the log partition function A in the exponential family. Recall in the discrete case the cost function analogue to the log scoring rule is $\mathsf{cost}(\mathbf{s})=b\ \log\sum_{\omega\in\Omega}e^{s_\omega/b}$. The continuous version of the log market scoring rule cost is:

$$\mathsf{A}(\boldsymbol{\theta})=\log\int_{\Omega^n}\mathsf{exp}\big(\langle\mathbf{x},\boldsymbol{\theta}\rangle\big)\nu(d\mathbf{x}).$$

This is in fact the log partition function. Now recall the probability or price quote in cost function based markets is $\nabla\mathsf{cost}(\boldsymbol{\theta})=\nabla\mathsf{A}(\boldsymbol{\theta})$. And by the moment matching conditions

of exponential family, the Jacobian of A (see $F.1$ and $F.3$ is the distribution's mean, vis $\nabla\mathsf{A}(\boldsymbol{\theta}) = \mathbf{E}_{\mathfrak{p}(\cdot;\boldsymbol{\theta}_\mu)}[\boldsymbol{\varphi}(\mathbf{x})] = \boldsymbol{\mu}$.

Hence there is a correspondence between eliciting the mean of the distribution and pricing the agent's contribution. However numerically speaking it is unclear how to use the elicited $\boldsymbol{\mu}$ in practice. In the case of Bernoulli variable on $\{0,1\}$, the mean is some value in $[0,1]$, which is easily interpreted as both likelihood and price. However in multivariate Gaussian the reported mean is $\Lambda = (\boldsymbol{\mu}, \boldsymbol{\sigma})$, which cannot be used as the price. Instead, Protocol 15 use the reported means to build the model $\mathfrak{p}(\mathbf{x};\boldsymbol{\mu})$, and then computes the likelihood of data under the model $\mathfrak{p}$. This dual market with state $\boldsymbol{\mu}$ has incentives:

$$\begin{aligned}
\rho(\boldsymbol{\mu}_\mathsf{k}, \boldsymbol{\mu}_{\mathsf{k}-1}; \mathsf{A}^*) &= -\Big(\mathsf{A}^*(\boldsymbol{\mu}_\mathsf{k}) - \mathsf{A}^*(\boldsymbol{\mu}_{\mathsf{k}-1}) - \nabla\mathsf{A}^*(\boldsymbol{\mu}_{\mathsf{k}-1})^\intercal(\boldsymbol{\mu}_\mathsf{k} - \boldsymbol{\mu}_{\mathsf{k}-1})\Big) \\
&= -\mathsf{D}(\boldsymbol{\mu}_{\mathsf{k}-1} || \boldsymbol{\mu}_\mathsf{k}) \\
&= -\int_{\Omega^n} \mathfrak{p}(\mathbf{x};\boldsymbol{\mu}_{\mathsf{k}-1})\ \log\frac{\mathfrak{p}(\mathbf{x};\boldsymbol{\mu}_{\mathsf{k}-1})}{\mathfrak{p}(\mathbf{x};\boldsymbol{\mu}_\mathsf{k})}\nu(d\mathbf{x}) \qquad \text{(D.11)} \\
\mathsf{score}(\mathbf{x}, \boldsymbol{\mu}_\mathsf{k}, \boldsymbol{\mu}_{\mathsf{k}-1}) &= \log(\mathfrak{p}(\mathbf{x};\boldsymbol{\mu}_\mathsf{k})) - \log(\mathfrak{p}(\mathbf{x};\boldsymbol{\mu}_{\mathsf{k}-1})) \\
&= \log\frac{\mathfrak{p}(\mathbf{x};\boldsymbol{\mu}_\mathsf{k})}{\mathfrak{p}(\mathbf{x};\boldsymbol{\mu}_{\mathsf{k}-1})}. \qquad \text{(D.12)}
\end{aligned}$$

Note the input to the score function is run on just one example of $\mathbf{x}$. Where as in familiar machine learning setting, one normally receive a test set of form $\mathbb{D}_\mathsf{test} = \{..\mathbf{x}_{test,\mathsf{i}}..\}$ with many examples. The Amm can simply run score for every $\mathbf{x}_{test,\mathsf{i}}$ in the test set, and sum the payouts/losses with: $\sum_s \mathsf{score}(\mathbf{x}_{test,\mathsf{i}}, \boldsymbol{\mu}_\mathsf{k}, \boldsymbol{\mu}_{\mathsf{k}-1})$ for every agent k. However this undermines the the max loss guarantee of expression $(D.9)$ whereby we assume only one final event is revealed and scored. This contradiction can be resolved by either scaling the price by a factor of up to $|\mathbb{D}_\mathsf{test}|$, or scale the final reward down by the same factor. The same measure can extend to models in production, where the model developer and data providers used to train the model are paid on a per-correct inference regime. What follows are a few examples where the distribution and the KL divergence can be written in closed form, so the sampling function $\mathbf{mc}$ in Protocol 15 is just the identity function that outputs the score function.

Example D.2.2 (Bernoulli bit)**.** There is some Bernoulli variable on $\{1,0\}$ with sufficient statistic $\varphi(x) = x$. The Amm elicits the likelihood of 1's. It initializes the market with $\mu_0 = 0.5$. The Amm sources data set and reveals training set. alice approaches the market, counts the training bits, and reports $\hat{\mu}_\mathsf{alice} = 0.6$. She is charged:

$$\rho(\mu_\mathsf{alice}, \mu_0) = 0.5 \cdot \ln\frac{0.5}{0.6} + 0.5 \cdot \ln\frac{0.5}{0.4} = \$0.0204.$$

The market ends and reveals test set $\{x_{test,\mathsf{i}}\} = \{0,0,1,1,1\}$. alice is paid for every bit she guesses correctly and penalized for the wrong bits:

$$\ln\left(\frac{0.4^2 \cdot 0.6^3}{0.5^5}\right) = 2 \cdot \ln\frac{0.4}{0.5} + 3 \cdot \ln\frac{0.6}{0.5} = \$0.10068.$$

We can scale up the each agent's ρ up to a factor of 5 to recover the Amm max loss guarantee.

The next two examples demonstrates two concepts:

1. how to write reward and price functions for conditional distributions;
2. sampling the data set to estimate KL divergence;

Example D.2.3. (*Logistic Regression*) In this case we are working with the conditional distribution. The data set is of form $\{..,(\mathbf{x}_\mathsf{i}, \mathsf{y}_\mathsf{i}),..\}$, where $\mathbf{x} \in \Omega^n$, and $y_\mathsf{i} \in \{1,0\}$. The goal is to predict $\mathfrak{p}(\mathsf{y}|\mathbf{x};\mathbf{w})$. Here we have some model $z = \mathbf{w}^\intercal\mathbf{x} + b$, where $\mathbf{w}$ is the weights vector,

and b is the bias term. We can lift both $\mathbf{w}$ and $\mathbf{x}$ into affine space and write $z = (\mathbf{w}\ b)^\intercal(\mathbf{x}\ 1)$ for cleaner notation, from now on it is understood that $\mathbf{w} = (\mathbf{w}\ b)$ and $\mathbf{x} = (\mathbf{x}\ 1)$. Next z is turned into a probability with $\boldsymbol{\sigma}(z) = \frac{1}{1+exp(-z)}$, so that the Bernoulli distribution of y conditioned on $\mathbf{x}$ is:

$$\mathfrak{p}(\mathsf{y}=1|\mathbf{x};\mathbf{w}) = \boldsymbol{\sigma}(\mathbf{w}^\intercal\mathbf{x}) = \frac{1}{1+e^{-\mathbf{w}^\intercal\mathbf{x}}} \qquad \mathfrak{p}(\mathsf{y}=0|\mathbf{x};\mathbf{w}) = \boldsymbol{\sigma}(1-\mathbf{w}^\intercal\mathbf{x}) = \frac{e^{-\mathbf{w}^\intercal\mathbf{x}}}{1+e^{-\mathbf{w}^\intercal\mathbf{x}}}.$$

Given some model parameterized by $\mathbf{w}$, the likelihood of some sample $(\mathbf{x}_\mathsf{i}, y_\mathsf{i})$ is:

$$\log(\mathfrak{p}(\mathsf{y}_\mathsf{i}|\mathbf{x}_\mathsf{i};\mathbf{w})) = \mathsf{y}_\mathsf{i}\log(\boldsymbol{\sigma}(\mathbf{w}^\intercal\mathbf{x}_\mathsf{i})) + (1-\mathsf{y}_\mathsf{i})\log(\boldsymbol{\sigma}(1-\mathbf{w}^\intercal\mathbf{x}_\mathsf{i})). \tag{D.13}$$

And note that when $y_\mathsf{i} = 0$, the expression above reduces to $(1-\mathsf{y}_\mathsf{i})\log(\boldsymbol{\sigma}(1-\mathbf{w}^\intercal\mathbf{x}_\mathsf{i}))$. When $\mathsf{y}_\mathsf{i} = 1$, expression D.13 becomes $\log(\boldsymbol{\sigma}(\mathbf{w}^\intercal\mathbf{x}_\mathsf{i}))$. The training objective given data set $\mathbb{D}_{\mathsf{train}} = \{..,(\mathbf{x}_\mathsf{i}, y_\mathsf{i}),..\}$ is:

$$\begin{aligned}&\arg\min_{\mathbf{w}\in\mathfrak{W}} \mathsf{loss}(\mathbf{w},\mathbf{w}_0) \ \text{ where}\\ &\mathsf{loss}(\mathbf{w},\mathbf{w}_0) = \sum_{(\mathbf{x}_\mathsf{i},y_\mathsf{i})\in\mathbb{D}_{\mathsf{train}}} -\mathsf{y}_\mathsf{i}\log(\boldsymbol{\sigma}(\mathbf{w}^\intercal\mathbf{x}_\mathsf{i})) - (1-\mathsf{y}_\mathsf{i})\log(\boldsymbol{\sigma}(1-\mathbf{w}^\intercal\mathbf{x}_\mathsf{i})).\end{aligned}$$

It can be shown that this expression is convex in $\mathbf{w}^\intercal\mathbf{x}$ [WJ08]. Now given some data point $(\mathbf{x}_\mathsf{i}, y_\mathsf{i})$, the reward of updating the model from $\mathbf{v}$ to $\mathbf{w}$ is:

$$\begin{aligned}\mathsf{score}((\mathbf{x}_\mathsf{i},y_\mathsf{i}),\mathbf{w},\mathbf{v}) &= \log(\mathfrak{p}(\mathsf{y}_\mathsf{i}|\mathbf{x}_\mathsf{i};\mathbf{w})) - \log(\mathfrak{p}(\mathsf{y}_\mathsf{i}|\mathbf{x}_\mathsf{i};\mathbf{v}))\\ &= \mathsf{y}_\mathsf{i}\cdot\log(\boldsymbol{\sigma}(\mathbf{w}^\intercal\mathbf{x}_\mathsf{i})) + (1-\mathsf{y}_\mathsf{i})\cdot\log(\boldsymbol{\sigma}(1-\mathbf{w}^\intercal\mathbf{x}_\mathsf{i}))\\ &\quad - \mathsf{y}_\mathsf{i}\cdot\log(\boldsymbol{\sigma}(\mathbf{v}^\intercal\mathbf{x}_\mathsf{i})) - (1-\mathsf{y}_\mathsf{i})\cdot\log(\boldsymbol{\sigma}(1-\mathbf{v}^\intercal\mathbf{x}_\mathsf{i})).\end{aligned} \tag{D.14}$$

Alternatively we can write eqn (D.14) as:

$$\mathsf{score}((\mathbf{x}_\mathsf{i},y_\mathsf{i}),\mathbf{w},\mathbf{v}) = \begin{cases}\log(\boldsymbol{\sigma}(1-\mathbf{w}^\intercal\mathbf{x}_\mathsf{i})) - \log(\boldsymbol{\sigma}(1-\mathbf{v}^\intercal\mathbf{x}_\mathsf{i})) & (y_\mathsf{i}=0)\\ \log(\boldsymbol{\sigma}(\mathbf{w}^\intercal\mathbf{x}_\mathsf{i})) - \log(\boldsymbol{\sigma}(\mathbf{v}^\intercal\mathbf{x}_\mathsf{i})) & (y_\mathsf{i}=1)\end{cases}$$

It can be shown that when $\mathbf{w}$ classifies $\mathbf{x}_\mathsf{i}$ more accurately than $\mathbf{v}$, then expression (D.14) is > 0, and it is < 0 otherwise. Notice the Agents are only graded on the part of the log-likelihood that estimates the probability of the given outcome y_i.

Now the market initializes $\mathbf{v} = \mathbf{0}$, so that $\mathfrak{p}(\mathsf{y}=1|\mathbf{x};\mathbf{v}) = \frac{1}{2}$. Then it publishes the training set $\mathbb{D}_{\mathsf{train}} = \{..(\mathbf{x}_s, y_\mathsf{i})...\}$. alice queries the training data, and submit $\mathbf{w}$. Her price evaluated at any given $(\mathbf{x}_\mathsf{i}, y_\mathsf{i})$ in the training data is:

$$\begin{aligned}-\rho((\mathbf{x}_\mathsf{i},y_\mathsf{i}),\mathbf{w},\mathbf{v}) &= \mathsf{D}(\mathfrak{p}(\mathsf{y}_\mathsf{i}|\mathbf{x}_\mathsf{i};\mathbf{v})||\mathfrak{p}(\mathsf{y}|\mathbf{x}_\mathsf{i};\mathbf{w}))\\ &= \mathbf{1}_{\mathsf{y}_\mathsf{i},1}\cdot\frac{1}{1+\exp(-\mathbf{v}^\intercal\mathbf{x}_\mathsf{i})}\cdot\ln\left(\frac{1+\exp(-\mathbf{w}^\intercal\mathbf{x}_\mathsf{i})}{1+\exp(-\mathbf{v}^\intercal\mathbf{x}_\mathsf{i})}\right)\\ &\quad + \ \mathbf{1}_{\mathsf{y}_\mathsf{i},0}\cdot\frac{\exp(-\mathbf{v}^\intercal\mathbf{x}_\mathsf{i})}{1+\exp(-\mathbf{v}^\intercal\mathbf{x}_\mathsf{i})}\cdot\ln\left(\frac{1+\exp(-\mathbf{v}^\intercal\mathbf{x}_\mathsf{i})}{\exp(-\mathbf{v}^\intercal\mathbf{x}_\mathsf{i})}\cdot\frac{\exp(-\mathbf{w}^\intercal\mathbf{x}_\mathsf{i})}{1+\exp(-\mathbf{w}^\intercal\mathbf{x}_\mathsf{i})}\right).\end{aligned}$$

Where $\mathbf{1}_{\mathsf{y}_\mathsf{i},1} = 1$ if $y_\mathsf{i} = 1$, and $\mathbf{1}_{\mathsf{y}_\mathsf{i},1} = 0$ otherwise. The price for the update is then computed over subset of the training set, named $\mathbb{D}_{\mathsf{train}}$:

$$\mathbf{mc}(\rho(\cdot,\mathbf{w},\mathbf{v}),\mathbb{D}_{\mathsf{train}}) = \frac{1}{|\mathbb{D}_{\mathsf{train}}|}\sum_{(\mathbf{x}_\mathsf{i},y_\mathsf{i})\in\mathbb{D}_{\mathsf{train}}}\mathsf{D}(\mathfrak{p}(y_\mathsf{i}|\mathbf{x}_\mathsf{i};\mathbf{v})||\mathfrak{p}(\mathsf{y}_\mathsf{i}|\mathbf{x}_\mathsf{i};\mathbf{w})).$$

Here **mc** uses the average KL divergence over all point-wise estimates on the training data. This average is the unbiased estimate of the true KL divergence. This expression is what alice will pay to update the model parameter from $\mathbf{v}$ to $\mathbf{w}$.

APPENDIX E

Machine Learning

E.1 Introduction

This chapter offers a brief introduction to the field of machine learning with a *narrow* pedagogical approach. In this view, the workhorse of machine learning is kernel regression. Here the data points are projected onto some "suitable space", and a surface is drawn to fit the points via function search. The next few sections make this intuition precise by:

1. expressing the properties of this Hilbert space $\mathcal{H}_\kappa$;
2. explore elementary regression techniques in $\mathcal{H}_\kappa$;
3. build towards feed-forward deep neural networks in context of this view.

This discourse is suitable for understanding QAudit, which is concerned with how to measure different models in a principled manner. Thus, many subjects are completely absent in this chapter. They include:

1. a basic survey into what machine learning is: how to train models, cross validation, and bias-variance trade-offs, etc.
2. A tour through the different machine learning "settings", i.e. generative vs discriminative models, reinforcement learning, transfer learning, and classical methods.
3. Any discussion of optimization techniques as seen in [BV04] and [GB16].
4. Practical concerns such as how to prepare data, train LLMs, or prompt them.

Also missing is a basic introduction into probability, although we do provide a primer on the exponential family distribution in Appendix F. All in all, this chapter is suitable for readers who already understand machine learning, and wish to judge whether Metheus takes the long view on the subject matter.

E.2 Hilbert Space

Definition

The nature Hilbert space $\mathcal{H}$ is defined by constraining the behavior of its *norm* and *distance metric*. Roughly speaking, distance in any well-formed $\mathcal{H}$ must correspond to intuitive sense of how the real 3D space we live in behaves. In particular, this intuition will translate to invariants around triangles and parallelograms.

Definition E.2.1. (Hilbert Space $\mathcal{H}$) Let $\mathbb{V}$ be a vector space. A norm is a mapping $\|\cdot\| : \mathbb{V} \to [0, \infty)$ that satisfies:

$$\begin{aligned}
&\|\mathbf{x}\| \geq 0 \ \text{ for every } \mathbf{x} \in \mathbb{V} && \text{(nonnegative)}\\
&\|\mathbf{x}\| = 0 \ \text{ iff } \mathbf{x} = 0 && \text{(strictly positive)}\\
&\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\| && \text{(triangle inequality)}\\
&\|a\mathbf{x}\| = a\|\mathbf{x}\| \text{ for every } a \in \mathbb{R} && \text{(homogeneous).}
\end{aligned}$$

In Hilbert spaces, the norm is expressed as the *inner product*, that is a mapping $\langle \cdot, \cdot \rangle : \mathbb{V} \times \mathbb{V} \to \mathbb{R}$ which satisfies for every $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathbb{V}$, and $a, b \in \mathbb{R}$:

$$\begin{aligned}
&\langle \mathbf{x}, \mathbf{x} \rangle \geq 0 && \text{(nonnegative)}\\
&\langle \mathbf{x}, \mathbf{x} \rangle = 0 \text{ iff } \mathbf{x} = 0 && \text{(strictly positive)}\\
&\langle \mathbf{x}, a\mathbf{y} + b\mathbf{z} \rangle = a\langle \mathbf{x}, \mathbf{y} \rangle + b\langle \mathbf{x}, \mathbf{z} \rangle && \text{(linear in the 2nd argument)}\\
&\langle \mathbf{x}, \mathbf{y} \rangle = \langle \mathbf{y}, \mathbf{x} \rangle && \text{(Hermitian symmetric).}
\end{aligned}$$

Vectors $\mathbf{x}$ and $\mathbf{y}$ are *orthogonal* if $\langle \mathbf{x}, \mathbf{y} \rangle = 0$. Now define:

$$\|\mathbf{x}\|_2 = \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle} \qquad \text{(norm)} \tag{E.1}$$

$$\|\mathbf{x}_1 - \mathbf{x}_2\|_2 = \sqrt{\langle \mathbf{x}_1, \mathbf{x}_2 \rangle} \qquad \text{(distance).} \tag{E.2}$$

And observe how the definition uses the L^2 norm, which resembles the Pythagorean's theorem. Finally, every Hilbert space must be complete. A space is complete if every *Cauchy sequence* converges to a limit, a sequence $\mathbf{x}_1, \mathbf{x}_2, \ldots$ is Cauchy if $\|\mathbf{x}_m - \mathbf{x}_n\|_2 \to 0$ as $m, n \to \infty$. A complete normed space is known as a *Banach space* [1] A *Hilbert Space* $\mathcal{H}$ then is a complete normed space where the norm is the L^2 inner product defined in eqn E.1 [[Bea73], [Axl20]].

The Reproducing-Kernel Hilbert Space (RKHS)

A RKHS is a class of smooth functions defined by the *Mercer Kernel* κ. Once a RKHS is defined, a learner can conduct "function search" in this space according to some cost criteria. For example, a reproducing kernel space could be defined by some kernel κ with a data set, and a learner would search this space for some linear function f that maximally separates the points. Mercer kernels are used when a learner wish to classify the data using a linear function, but the data is not separable by it (see fig (E.1)). One solution is to "lift" the data to a higher dimension using a kernel κ, and search for a linear separator in the new space (so called "feature space") induced by κ (see fig (E.2)).

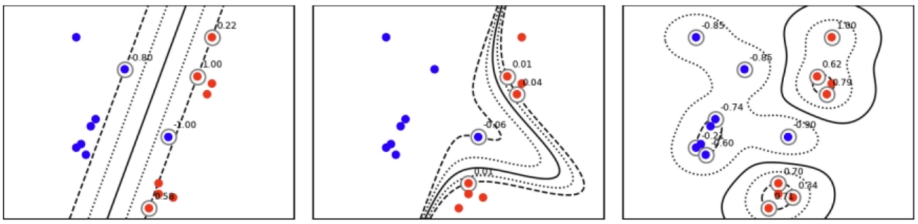


Figure E.1: Left: original data space where a linear classifier cannot shatter the data set, as evidence by the points circled (best seen in color). Center: a polynomial kernel that describes a quadratic separator, which is linear in feature space, but is nonlinear when projected back into data space. Right: radial basis function (RBF) kernel that is even more expressive: the classification boundaries become discrete islands when projected into data space [Van23].

[1] Note all Hilbert spaces are Banach spaces, but the converse is not true.

Remark E.2.2. Linear solutions are "nice" because they are less expensive to train, and have convex loss functions, which ensures the solution they find is optimal. Another avenue is to use nonlinear expressions to separate the data points, as is the case with artificial neural networks do (ANN). ANNs and kernel-based methods are different ways of addressing the same problem: classifying points that cannot be shattered by linear separators. However where as the kernels κ have to be hand designed, ANN learn the feature space directly from data. Every mapping is still some polynomial, however the learned polynomial are so non-intuitive no person could have derived it from either principle or experience. Nonetheless, understanding Kernels inform the understanding of ANNs.

Definition E.2.3. (**Mercer Kernel** κ). A Mercer kernel is a continuous function κ that is *symmetric* and *positive semidefinite*, that is for every $f \in \mathcal{H}$, and $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{X}$:

$$\begin{aligned}
&\kappa : \mathbb{X} \times \mathbb{X} \to \mathbb{R} \\
&\kappa(\mathbf{x}_1, \mathbf{x}_2) = \kappa(\mathbf{x}_2, \mathbf{x}_1) && \text{(symmetric)} \\
&\int_{\mathbb{X}} \int_{\mathbb{X}} \kappa(\mathbf{x}_1, \mathbf{x}_2) f(\mathbf{x}_1) f(\mathbf{x}_2) d\mathbf{x}_1 d\mathbf{x}_2 \geq 0 && \text{(positive semidefinite).}
\end{aligned}$$

These two conditions ensure κ is a proper distance, or "similarity" relation. An example of κ is the Gaussian radial basis function (RBF) kernel:

$$\kappa(\mathbf{x}_1, \mathbf{x}_2) = \exp\left(-\frac{\|\mathbf{x}_1 - \mathbf{x}_2\|_2^2}{2\sigma^2}\right) \qquad \text{(RBF kernel).} \tag{E.3}$$

If we partially apply κ so that one of the argument is fixed, then this partially applied function is the **feature map**: [2]

$$\begin{aligned}
&\phi_{\mathbf{x}_1} : \mathbb{X} \to \mathbb{R} \\
&\phi_{\mathbf{x}_1} := \kappa(\mathbf{x}_1, \cdot).
\end{aligned}$$

And notice the feature map is a vector, and the notation $\phi_{\mathbf{x}}$ evokes the basis of eqn (**??**). The set of feature vectors will span all possible functions in the Kernel space. For example in the RBF kernel, if we let $\mathbf{x}_1 = 3$, then we have: $\phi_3(\mathbf{x}_2) = \exp\left(\frac{\|3-\mathbf{x}_2\|_2^2}{2\sigma^2}\right)$. The inner product of two partially applied kernels is defined with:

$$\langle \phi_{\mathbf{x}_1}, \phi_{\mathbf{x}_2} \rangle = \underbrace{\langle \kappa(\mathbf{x}_1, \cdot), \kappa(\mathbf{x}_2, \cdot) \rangle = \kappa(\mathbf{x}_1, \mathbf{x}_2) = \phi_{\mathbf{x}_1}^\intercal \phi_{\mathbf{x}_2}}_{\text{reproducing property}}. \tag{E.4}$$

The result will be a scalar expressing similarity of $\mathbf{x}_1$ to $\mathbf{x}_2$ in the space defined by kernel κ. Observe this expression uses the reproducing property of κ, whereby when two partially applied functions are applied to each other, it is just the inner product [[Was19], [SC04]].

E.3 Learning in Hilbert Space

Note that it is not the case where given some data set, and arbitrary training objective, the solution to this objective must be found in the $\mathcal{H}_\kappa$. The next theorem shows which kind of objective functions yield solutions in the RKHS defined by the kernel.

Theorem E.3.1. ***(The Representer Theorem).*** *Let the data set be* $\{(\mathbf{x}_i, \mathsf{y}_i) : i = 1..m\}$. *Let* κ *be a reproducing kernel function, so that there is a RKHS* $\mathcal{H}_\kappa$ *spanned by the* κ*-embedding of data set:* $\{\kappa(\mathbf{x}_i, \cdot) : i = 1..m\}$ *(see fig (E.3)). Now suppose the data is generated by function*

[2] Sometimes this notation $\phi(\mathbf{x}_1)$ is used to denote $\phi_{\mathbf{x}_1}$. In this case $\phi(\mathbf{x}_1)(\mathbf{x}_2) = \langle \phi_{\mathbf{x}_1}, \phi_{\mathbf{x}_2} \rangle$.

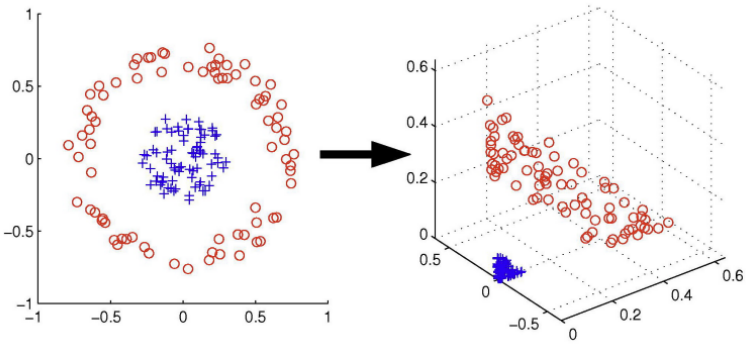


Figure E.2: Left: Given m data points $\mathbf{X} \in \mathbb{R}^{\mathsf{m}\times 2}$, where each data point $\mathbf{x} = (a_1, a_2) \in \mathbb{R}^2$, $\mathbf{X}$ is not separable by linear classifiers. Right: the data set is "lifted" into a higher dimension by the kernel, where each point in this new space is defined $\kappa(\mathbf{x}, \cdot) = \phi_{\mathbf{x}} = (a_1^2, a_2^2, \sqrt{2}\, a_1 a_2)$. This set of points is now separated by a hyperplane in $\mathbb{R}^3$ [TJ04].

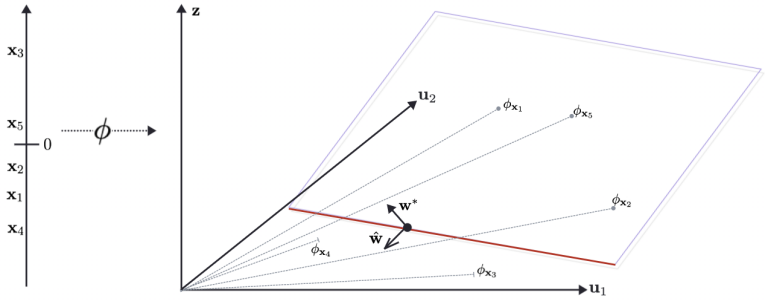


Figure E.3: The labeled data set $\{(\mathbf{x}_1, 1), (\mathbf{x}_2, 1), (\mathbf{x}_3, -1), (\mathbf{x}_4, -1), (\mathbf{x}_5, 1)\}$ is not separable in the original data space in $\mathbb{R}$. They are projected onto Hilbert space $\mathcal{H}_\kappa$ in $\mathbb{R}^2$, with kernel κ and feature map ϕ. $\mathcal{H}_\kappa$ is spanned by the kernalized data $\{\phi_{\mathbf{x}_i} = \kappa(\mathbf{x}_i, \cdot)\}_{i=1}^5$, or the orthonormal basis $\{\mathbf{u}_1, \mathbf{u}_2\}$. $\mathcal{H}_\kappa$ rests in ambient space in $\mathbb{R}^3$ spanned by $\{\mathbf{u}, \mathbf{u}_2, \mathbf{z}\}$. Representer theorem states that if the learning objective is defined as eqn (E.5), then any solution $\mathbf{w}^* = \hat{\mathbf{w}} + \mathbf{w}_\perp$ in $\mathbb{R}^3$ has a component $\hat{\mathbf{w}}$ lying in $\mathcal{H}_\kappa$ (in $\mathbb{R}^2$), which also shatters the points. That is $\hat{\mathbf{w}}$ is spanned by the feature vectors with: $\hat{\mathbf{w}} = \sum_{i=1}^5 \alpha_i \phi_{\mathbf{x}_i}$.

$\mathsf{y}_i = f(\mathbf{x}_i) = \mathbf{x}_i{}^\intercal \mathbf{w}$. *Define regularizer* Γ *that is non-decreasing, and estimator* $\mathsf{y}_i = \mathbf{w}^\intercal \mathbf{x}_i$. *If the original objective is of form:*

$$\arg\min_{\mathbf{w}} \; loss(\mathbf{w}) \qquad where \qquad loss(\mathbf{w}) = \underbrace{\sum_{i=1}^{m} (\mathsf{y}_i - \langle \mathbf{w}, \mathbf{x}_i \rangle)^2}_{prediction\ loss} + \underbrace{\Gamma(\|\mathbf{w}\|_2^2)}_{regularizer}. \tag{E.5}$$

Then a solution $\hat{\mathbf{w}}$ *lies in* $\mathcal{H}_\kappa$, *that is* $\hat{\mathbf{w}}$ *can be described as a linear combination of the kernalized data set with:*

$$\hat{\mathbf{w}} = \sum_{i=1}^{m} \alpha_i \kappa(\mathbf{x}_i, \cdot). \tag{E.6}$$

Proof. Suppose there is some solution $\mathbf{w}^* = \hat{\mathbf{w}} + \mathbf{w}_\perp$, where $\hat{\mathbf{w}}$ lies in $\mathcal{H}_\kappa$, and $\mathbf{w}_\perp$ lies in

$\mathcal{H}_\kappa^\perp$, that is the space orthogonal to $\mathcal{H}_\kappa$. [3] First we show $\hat{\mathbf{w}}$ minimizes the regularizer:

$$\|\mathbf{w}^*\|_2^2 = \|\hat{\mathbf{w}}\|_2^2 + \|\mathbf{w}_\perp\|_2^2 \geq \|\hat{\mathbf{w}}\|_2^2.$$

And since Γ is non decreasing, we know:

$$\Gamma(\|\mathbf{w}^*\|_2^2) = \Gamma(\|\hat{\mathbf{w}}\|_2^2 + \|\mathbf{w}_\perp\|_2^2) \geq \Gamma(\|\hat{\mathbf{w}}\|_2^2).$$

This means the regularizer Γ can be minimized by the $\hat{\mathbf{w}}$ lying in $\mathcal{H}_\kappa$. Now we show the classification loss only depend on the component lying on the subspace $\mathcal{H}_\kappa$. For every $\mathbf{x}_\mathsf{i}$ evaluated in the kernalized space:

$$\langle \mathbf{w}^*, \kappa(\mathbf{x}_\mathsf{i}, \cdot)\rangle = \langle \hat{\mathbf{w}}, \kappa(\mathbf{x}_\mathsf{i}, \cdot)\rangle + \underbrace{\langle \mathbf{w}_\perp, \kappa(\mathbf{x}_\mathsf{i}, \cdot)\rangle}_{0 \text{ by def}} = \langle \hat{\mathbf{w}}, \kappa(\mathbf{x}_\mathsf{i}, \cdot)\rangle.$$

So the part of $\mathbf{w}^*$ that is orthogonal to $\mathcal{H}_\kappa$ does not influence the optimization objective. Hence we know $\hat{\mathbf{w}} = \sum_{\mathsf{i}=1}^{\mathsf{m}} \alpha_\mathsf{i}\kappa(\mathbf{x}_\mathsf{i}, \cdot)$ [Bar08]. ■

Since support vector machines (SVM) and Ridge regression have the loss function stated above, they can be *kernalized*. That is to say it is possible to learn an SVM classifier or run kernalized ridge regression, in the feature space defined by κ. The following example demonstrates a simple kernel regression, and introduces the concept of Kernel matrix or Gram matrix.

Example E.3.2. (Kernalized Ridge Regression)
Given a set of data of form $\{(\mathbf{x}_\mathsf{i}, \mathsf{y}_\mathsf{i}) : \mathsf{i} = 1..\mathsf{m}\}$, where each $\mathbf{x} \in \mathbb{R}^{\mathsf{d}_0 \times 1}$, and the label vector $\mathbf{y} = (\mathsf{y}_1, .., \mathsf{y}_\mathsf{m}) \in \mathbb{R}^{1\times \mathsf{m}}$. Assume the data is generated by some function $\mathsf{y}_\mathsf{i} = f(\mathbf{x}_\mathsf{i}) = \mathbf{w}^\intercal \mathbf{x}_\mathsf{i}$, then $\mathbf{w}$ is estimated by minimizing the regularized *quadratic loss* in the data space with:

$$loss(\mathbf{w}) = \frac{1}{2}\sum_{\mathsf{i}=1}^{\mathsf{m}} (\mathsf{y}_\mathsf{i} - \langle \mathbf{w}, \mathbf{x}_\mathsf{i}\rangle)^2 + \lambda\|\mathbf{w}\|_2^2$$

In the kernalized space, this loss becomes:

$$\begin{aligned} loss(\mathbf{w}) &= \frac{1}{2}\sum_{\mathsf{i}=1}^{\mathsf{m}} (\mathsf{y}_\mathsf{i} - \langle \mathbf{w}, \phi_{\mathbf{x}_\mathsf{i}}\rangle)^2 + \lambda\|\mathbf{w}\|_2^2 \\ &\overset{(1)}{=} \underbrace{\frac{1}{2}\sum_{\mathsf{i}=1}^{\mathsf{m}} \left(\mathsf{y}_\mathsf{i} - \left\langle \sum_{\mathsf{j}=1}^{\mathsf{m}} \alpha_\mathsf{j}\phi_{\mathbf{x}_\mathsf{j}},\ \phi_{\mathbf{x}_\mathsf{i}} \right\rangle\right)^2}_{\text{prediction loss}} + \underbrace{\lambda\|\mathbf{w}\|_2^2}_{\text{regularization}} \end{aligned}$$

Where going into (1) we use the fact that $\mathbf{w}$ is spanned by the feature vectors via: $\mathbf{w} = \sum_{\mathsf{j}=1}^{\mathsf{m}} \alpha_\mathsf{j}\phi_{\mathbf{x}_\mathsf{j}}$. Now rewrite "prediction loss" for $\mathbf{x}_\mathsf{i}$ w.r.t all feature basis with:

$$\sum_{\mathsf{j}=1}^{\mathsf{m}} \alpha_\mathsf{j}\langle \phi_{\mathbf{x}_\mathsf{j}}, \phi_{\mathbf{x}_\mathsf{i}}\rangle = (\alpha_1, \ldots, \alpha_\mathsf{m})(\kappa(\mathbf{x}_1, \mathbf{x}_\mathsf{i}), \ldots, \kappa(\mathbf{x}_\mathsf{m}, \mathbf{x}_\mathsf{i}))^\intercal = \langle \boldsymbol{\alpha}, \mathbf{k}_\mathsf{i}\rangle$$

We can place all $\mathbf{k}_\mathsf{i}$'s into an $\mathsf{m} \times \mathsf{m}$ matrix with:

$$\mathbf{K} = [\mathbf{k}_1, \ldots, \mathbf{k}_\mathsf{m}] = \begin{bmatrix} \kappa(\mathbf{x}_1, \mathbf{x}_1) & \ldots & \kappa(\mathbf{x}_1, \mathbf{x}_\mathsf{m}) \\ \vdots & \ddots & \vdots \\ \kappa(\mathbf{x}_\mathsf{m}, \mathbf{x}_1) & \ldots & \kappa(\mathbf{x}_\mathsf{m}, \mathbf{x}_\mathsf{m}) \end{bmatrix} \tag{E.7}$$

[3] I.e.: $\mathbf{w}^* = (a, b, c)$, $\hat{\mathbf{w}} = (a, b, 0)$, and $\mathbf{w}_\perp = (0, 0, c)$. And with direct sum we have: $\mathbf{w}^* = \hat{\mathbf{w}} + \mathbf{w}_\perp$.

This $\mathbf{K}$ is known as the Gram matrix, the *Kernel matrix*. It is the principle object of study in all Kernel methods (more on this later). The loss can be rewritten in vector notation using $\mathbf{K}$:

$$\frac{1}{2}\sum_{i=1}^{m}\left(y_i - \left\langle \sum_{j=1}^{m}\alpha_j\phi_{\mathbf{x}_j},\ \phi_{\mathbf{x}_i}\right\rangle\right)^2 = \frac{1}{2}\|\mathbf{y}-\mathbf{K}\boldsymbol{\alpha}\|_2^2 = \frac{1}{2}(\mathbf{y}-\mathbf{K}\boldsymbol{\alpha})^\intercal(\mathbf{y}-\mathbf{K}\boldsymbol{\alpha}).$$

And similarly for the regularizer:

$$\begin{aligned}\|\mathbf{w}\|_2^2 = \langle \mathbf{w},\mathbf{w}\rangle &= \langle \sum_{i=1}^{m}\alpha_i\phi_{\mathbf{x}_i}, \sum_{j=1}^{m}\alpha_j\phi_{\mathbf{x}_j}\rangle \\ &= \sum_{i,j}^{m}\alpha_i\alpha_j\langle\kappa(\mathbf{x}_i,\cdot),\kappa(\mathbf{x}_j,\cdot)\rangle = \sum_{i,j}^{m}\alpha_i\alpha_j\kappa(\mathbf{x}_i,\mathbf{x}_j) = \boldsymbol{\alpha}^\intercal\mathbf{K}\boldsymbol{\alpha}.\end{aligned}$$

And the objective becomes:

$$\mathbf{w}^* = \underset{\mathbf{w}\in\mathcal{H}_\kappa}{\arg\min}\ \frac{1}{2}\underbrace{(\mathbf{y}-\mathbf{K}\boldsymbol{\alpha})^\intercal(\mathbf{y}-\mathbf{K}\boldsymbol{\alpha})}_{\text{loss function}} + \underbrace{\lambda\boldsymbol{\alpha}^\intercal\mathbf{K}\boldsymbol{\alpha}.}_{\text{regularization}} \tag{E.8}$$

Solving for $\boldsymbol{\alpha}$ and we have:

$$\mathbf{w}^* = \sum_{i}^{m}\alpha_i\phi_{\mathbf{x}_i} = \mathbf{K}(\mathbf{K}+\lambda\mathbf{I})^{-1}\mathbf{y} \qquad \text{where } \boldsymbol{\alpha} = (\mathbf{K}+\lambda\mathbf{I})^{-1}\mathbf{y}. \tag{E.9}$$

And note that $\mathbf{w}^*$ is in the span of data set in feature space. And $\boldsymbol{\alpha}$ is informed by the Kernel matrix $\mathbf{K}$, and biased by the regularization constant λ. Moreover, the regularization is added to the diagonal element of $\mathbf{K}$ with $\mathbf{K}+\lambda\mathbf{I}$. In the case of classification via i.e. SVMs, adding λ's to the diagonal terms creates a greater margin on the classification boundary. Finally, observe the feature vectors $\phi_{\mathbf{x}_i}$ are not needed for regression, all necessary information about the data is stored in the Gram matrix. This makes $\mathbf{K}$ the principle object of study across Kernel methods [[Wel19], [Sch00], [Rad+22a]].

Kernel matrix $\mathbf{K}$

The $m \times m$ square matrix $\mathbf{K}$ in eqn (E.7) is the central *"data structure"* of interest in any kernel-based method. Each one of its ijth entries is defined with the inner product of the embedding of the ith data point with that of the jth one:

$$\mathbf{K}_{ij} = \kappa(\mathbf{x}_i,\mathbf{x}_j) = \langle\phi_{\mathbf{x}_i},\phi_{\mathbf{x}_j}\rangle \qquad \forall\ 1\le i,j\le m. \tag{E.10}$$

Some basic observations follows.

1. $\mathbf{K}$ *is an information bottleneck*, interfacing between data on one side, and learning algorithm on the other. Many of the information is lost to the learning algorithms, i.e. the Kernel is invariant to where the origin is placed in $\mathcal{H}_\kappa$ [SC04, p. 69].

2. Since κ is a Mercer kernel by definition, $\mathbf{K}$ is symmetric and positive semidefinite:

$$\begin{aligned}&\mathbf{K} = \mathbf{K}^\intercal && \text{(symmetric)}\\ &\mathbf{x}^\intercal\mathbf{K}\mathbf{x} \succeq 0 && \text{(positive semidefinite).}\end{aligned}$$

   Notice the two ways of defining symmetric and nonnegative conditions are analogous.

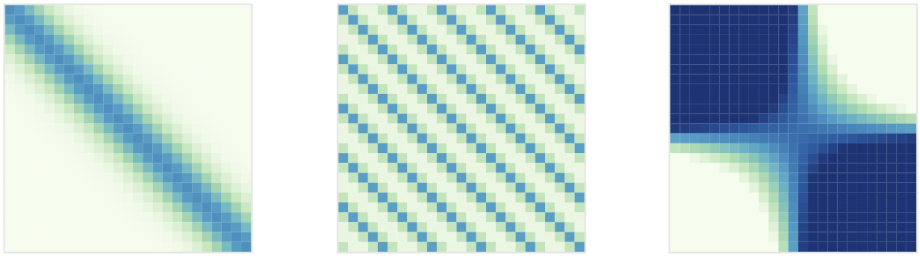


Figure E.4: Placing existing kernels κ into polynomials or exponential functions create new valid kernels. These kernels induce different kernel matrices $\mathbf{K}$, with different similarity scores between any pair of $\mathbf{x}_1$ and $\mathbf{x}_2$.

A) RBF/Gaussian kernel $\kappa(\mathbf{x}_1, \mathbf{x}_2) = \exp\left(-\frac{\|\mathbf{x}_1 - \mathbf{x}_2\|_2^2}{2\sigma^2}\right)$.

B) periodic kernel $\kappa(\mathbf{x}_1, \mathbf{x}_2) = \sigma^2 \exp\left(-\frac{2sin^2(\pi|\mathbf{x}_1 - \mathbf{x}_2|/p)}{2\sigma^2}\right)$.

C) linear kernel $\kappa(\mathbf{x}_1, \mathbf{x}_2) = \sigma_1^2 + \sigma_2^2(\mathbf{x}_1 - b)(\mathbf{x}_2 - b)$.

Darker colors indicate higher similarity scores, notice in each case the diagonal represent self similarity. In every case, the variance parameter σ determines the average distance away from the function's mean when running regression on $\mathbf{K}$ [GKD19].

3. *Kernel trick*: this is an optimization choice, where the entire Gram matrix $\mathbf{K}$ is pre-computed and stored for reference. For example, if κ is the RBF kernel defined in eqn $(E.3)$, then its feature vector at each $\mathbf{x}_\mathsf{i}$ is:

$$\phi_{\mathbf{x}_\mathsf{i}} = \kappa(\mathbf{x}_\mathsf{i}, x) = \exp\left(-\frac{\|\mathbf{x}_\mathsf{i} - x\|_2^2}{2\sigma^2}\right).$$

Now because $e^x = \sum_k^\infty 1/(k!)x^k$ is a polynomial of an infinite degree (or a very high degree on a computer), this feature vector is potentially very large degree d. Thus if the dimensionality of the feature vector d is much larger than the dimensionality of the data m, then it makes sense to pre-compute the Gram matrix of size $\mathsf{m} \times \mathsf{m}$. So that given $\mathbf{x}_\mathsf{i}, \mathbf{x}_\mathsf{j} \in \{(\mathbf{x}_\mathsf{i}, \mathsf{y}_\mathsf{i}) : \mathsf{i} = 1..\mathsf{m}\}$, the fully computed kernel is just a scalar:

$$\kappa(\mathbf{x}_\mathsf{i}, \mathbf{x}_\mathsf{j}) = \exp\left(-\frac{\|\mathbf{x}_\mathsf{i} - \mathbf{x}_\mathsf{j}\|_2^2}{2\sigma^2}\right) = \exp\left(-\frac{\mathbf{x}_\mathsf{i}^\intercal \mathbf{x}_\mathsf{j}}{2\sigma^2}\right).$$

And the $d \times 1$ feature vector $\phi_{\mathbf{x}_\mathsf{i}}$ is not needed, this is the trick. In the case where the $m^2 < m \times d$, computing the Gram matrix is efficient. However in the case where $m^2 > m \times d$, then pre-computing the Gram matrix could be very expensive space wise [[Sch00], [Rad+22a]]. [4]

4. *Eigenvalue decomposition of* $\mathbf{K}$: since $\mathbf{K}$ is square, it can be decomposed into:

$$\mathbf{K} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\intercal = \sum_{\mathsf{i}=1}^{\mathsf{m}} \lambda_\mathsf{i} \mathbf{v}_\mathsf{i} \mathbf{v}_\mathsf{i}^\intercal, \tag{E.11}$$

[4] For the programming inclined, the feature vector $\kappa(\mathbf{x}_\mathsf{i}, \cdot) = \phi_{\mathbf{x}_\mathsf{i}}$ is similar to partially evaluated function in Lazy programming languages, where i.e. summation over a list is only evaluated "half way." So that the spine of the list is opened, but the summation has not been carried through. Depending on the size of d, this could cause a space leak. Where as the fully applied kernel $\kappa(\mathbf{x}_\mathsf{i}, \mathbf{x}_\mathsf{j})$ is equivalent to forcing the summation to finish with the summed number.

where λ_i and $\mathbf{v}_i$ are eigenvalues and eigenvectors of $\mathbf{K}$. The set of $\mathbf{v}_i$'s define the feature space spanned by the training set of $\mathbf{x}$'s. In fig (E.3), the basis set $\{\mathbf{u}_1, \mathbf{u}_2\}$ can be found by eigenvalue decomposition.

Example E.3.3. (Kernel Logistic Regression) We can apply the kernel closure properties in this *classification* model. [5] Given data set of m labeled points: $\mathbf{X} \in \mathbb{R}^{\mathsf{m} \times \mathsf{d}_0}$, with $\mathbf{x} \in \mathbb{R}^{\mathsf{d}_0}$, and labels $\mathbf{y} \in \{0,1\}^{\mathsf{m}}$. The model estimates the likelihood of label y given $\mathbf{x}$ with:

$$\begin{aligned} \mathfrak{p}(\mathsf{y}=1|\mathbf{x};\mathbf{w}) &= \boldsymbol{\sigma}(\mathbf{w}^\intercal \mathbf{x}) = \frac{1}{1+\exp(-\mathbf{w}^\intercal \mathbf{x})}, \\ \mathfrak{p}(\mathsf{y}=0|\mathbf{x};\mathbf{w}) &= \boldsymbol{\sigma}(1-\mathbf{w}^\intercal \mathbf{x}) = \frac{\exp(-\mathbf{w}^\intercal \mathbf{x})}{1+\exp(-\mathbf{w}^\intercal \mathbf{x})}. \end{aligned} \tag{E.12}$$

Where $\boldsymbol{\sigma}(\cdot)$ is the *sigmoid activation function*, defined as: $\boldsymbol{\sigma}(x) = 1/(1+e^{-x})$, and the input to $\boldsymbol{\sigma}(\cdot)$ is often called the *logit*. As an aside, the bias term w_o is represented by lifting the data into affine space with: $\mathbf{w} = (\mathbf{w}, w_o)$ and $\mathbf{x} = (\mathbf{x}, 1)$, so that $\mathbf{w}^\intercal \mathbf{x} = \mathbf{w}^\intercal \mathbf{x} + w_o$.

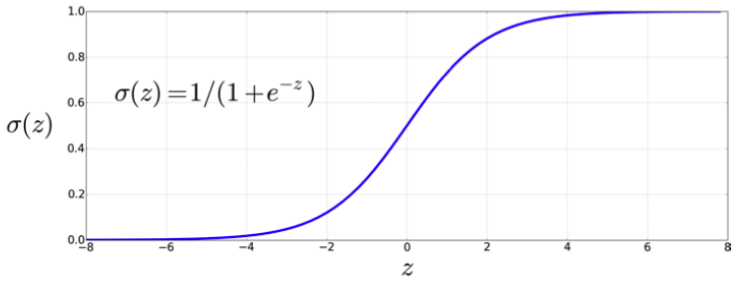


Figure E.5: The sigmoid activation function maps real values onto probabilities. In this case, the logit is $z = \langle \mathbf{w}, \mathbf{x} \rangle$. Note for a function to be convex, the epigraph of the function must be above the tangent plane supporting every point in this function. Hence $\boldsymbol{\sigma}(\cdot)$ is not a convex function. However, the overall learning objective can still be convex, provided the relationship between log-loss and the model parameter $\mathbf{w}$ is appropriately expressed [JM24].

The model parameters can be found by maximum likelihood estimation (MLE), where the learning objective find the $\mathbf{w}$ that maximizes the likelihood of the training data. So given any example $(\mathbf{x}_i, \mathsf{y}_i)$, its likelihood under eqn (E.12) is:

$$\begin{aligned} \mathfrak{p}(\mathsf{y}_i|\mathbf{x}_i;\mathbf{w}) &= \mathfrak{p}(\mathsf{y}=1|\mathbf{x};\mathbf{w})^{\mathsf{y}_i} \cdot \mathfrak{p}(\mathsf{y}=0|\mathbf{x};\mathbf{w})^{(1-\mathsf{y}_i)} \\ &= (\boldsymbol{\sigma}(\mathbf{w}^\intercal \mathbf{x}))^{\mathsf{y}_i} \cdot (\boldsymbol{\sigma}(1-\mathbf{w}^\intercal \mathbf{x}))^{(1-\mathsf{y}_i)}. \end{aligned}$$

Where y_i is the given label or ground truth, while $\mathfrak{p}(\mathsf{y}_i|\mathbf{x}_i;\mathbf{w})$ is the likelihood of the label y_i occurring given the value of $\mathbf{x}_i$, under the model parameterized by $\mathbf{w}$. Observe how if $\mathsf{y}_i = 1$, then the likelihood reduces to $\mathfrak{p}(\mathsf{y}_i|\mathbf{x}_i;\mathbf{w}) = \boldsymbol{\sigma}(\mathbf{w}^\intercal \mathbf{x})$, and if $\mathsf{y}_{i=0}$, then the expression is: $\mathfrak{p}(\mathsf{y}_i|\mathbf{x}_i;\mathbf{w}) = \boldsymbol{\sigma}(1-\mathbf{w}^\intercal \mathbf{x})$. The log-likelihood of a singled labeled data pair is then:

$$\log\big(\mathfrak{p}(\mathsf{y}_i|\mathbf{x}_i;\mathbf{w})\big) = \mathsf{y}_i \log\big(\boldsymbol{\sigma}(\mathbf{w}^\intercal \mathbf{x})\big) + (1-\mathsf{y}_i)\log\big(\boldsymbol{\sigma}(1-\mathbf{w}^\intercal \mathbf{x})\big).$$

Log likelihood is used to address the numerical underflow problem. Since the likelihood of the entire data set would grow vanishingly small, if the *data set* likelihood is expressed

[5] Even though this is called "regression."

with: $\prod_i^{\mathsf{m}} \mathfrak{p}(\mathsf{y}_{\mathsf{i}=1|\mathbf{x}_\mathsf{i};\mathbf{w})^{\mathsf{y}_\mathsf{i}} \mathfrak{p}(\mathsf{y}_{\mathsf{i}=0|\mathbf{x}_\mathsf{i};\mathbf{w})^{1-\mathsf{y}_\mathsf{i}}}$. Thus, the objective is to maximize the log-likelihood of data, or minimize the negative loss over all m examples with:

$$\log\big(loss(\mathbf{w})\big) = -\sum_{\mathsf{i}=1}^{\mathsf{m}} \Big(\mathsf{y}_\mathsf{i}\log\big(\boldsymbol{\sigma}(\mathbf{w}^\intercal\mathbf{x}_\mathsf{i})\big) + (1-\mathsf{y}_\mathsf{i})\log\big(\boldsymbol{\sigma}(1-\mathbf{w}^\intercal\mathbf{x}_\mathsf{i})\big)\Big). \tag{E.13}$$

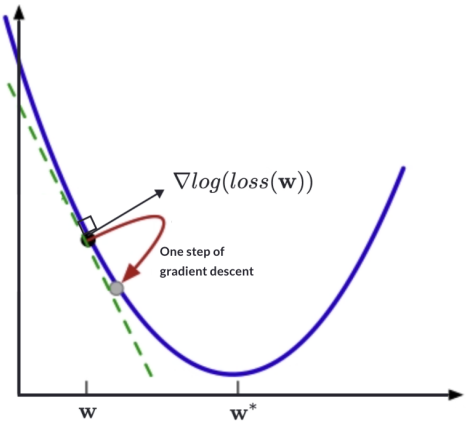


Figure E.6: In gradient descent, there is a convex relationship between the log-loss and model parameter $\mathbf{w}$. At any $\mathbf{w}$, we can compute the gradient or slope of the curve with $\nabla log(loss(\mathbf{w}))$. Because the curve is convex, there is a unique optimal solution at $\mathbf{w}^*$. The solver can find this solution provided the step size α is sufficiently small [JM24].

And note that because log is a convex function, its convex combination is convex. However unlike ridge regression, this objective does not admit a closed form solution. Instead gradient descent is used to update the $\mathbf{w}$ with:

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \alpha\nabla\log\big(loss(\mathbf{w})\big),$$

where $\alpha \in (0,1)$ is the learning rate. Next we can kernalize eqn $(E.12)$ with:

$$\begin{aligned}\mathfrak{p}(\mathsf{y}=1|\mathbf{x};\mathbf{w}) &= \boldsymbol{\sigma}(\mathbf{w}^\intercal\phi_\mathbf{x}) = \frac{1}{1+\exp(-\mathbf{w}^\intercal\phi_\mathbf{x})}, \\ \mathfrak{p}(\mathsf{y}=0|\mathbf{x};\mathbf{w}) &= \boldsymbol{\sigma}(1-\mathbf{w}^\intercal\phi_\mathbf{x}) = \frac{\exp(-\mathbf{w}^\intercal\phi_\mathbf{x})}{1+\exp(-\mathbf{w}^\intercal\phi_\mathbf{x})}.\end{aligned} \tag{E.14}$$

And note that by the closure properties defined above, $\mathfrak{p}(\mathsf{y}|\phi_\mathbf{x};\mathbf{w})$ is a valid kernel. Where given data point $\mathbf{x}_\mathsf{j}$, and kernel matrix $\mathbf{K}$ with entries $\kappa(\mathbf{x}_\mathsf{i},\mathbf{x}_\mathsf{j})$, we have, i.e.:

$$\mathfrak{p}(\mathsf{y}_\mathsf{j}=1|\mathbf{x}_\mathsf{j};\mathbf{w}) = \frac{1}{1+\exp\Big(-\sum_{\mathsf{i}=1}^{\mathsf{m}}\alpha_\mathsf{i}\kappa(\mathbf{x}_\mathsf{i},\mathbf{x}_\mathsf{j})\Big)}.$$

So that the kernel matrix $\mathbf{K}$ is used here during training and inference. The loss function then becomes:

$$\log\big(loss(\mathbf{w})\big) = -\sum_{\mathsf{i}=1}^{\mathsf{m}} \Big(\mathsf{y}_\mathsf{i}\log\big(\boldsymbol{\sigma}(\mathbf{w}^\intercal\phi_{\mathbf{x}_\mathsf{i}})\big) + (1-\mathsf{y}_\mathsf{i})\log\big(\boldsymbol{\sigma}(1-\mathbf{w}^\intercal\phi_{\mathbf{x}_\mathsf{i}})\big)\Big). \tag{E.15}$$

This *loss* can be optimized by the same numerical method outlined above [JM24].

Remark E.3.4. (**Maximizing likelihood vs minimizing KL divergence**) In the MLE objective above, we maximize the likelihood of the model given data. So that given current model settings $\mathfrak{p}(y|\mathbf{x};\mathbf{w})$, and optimal solution $\mathfrak{p}(y|\mathbf{x};\mathbf{w}^*)$, we can write:

$$\begin{aligned}
&\arg\max_{\mathbf{w}} \prod_{i=1}^{m} \mathfrak{p}(y_i|\mathbf{x}_i;\mathbf{w}) \\
&\qquad \implies \arg\max_{\mathbf{w}} \frac{m}{m} \cdot \sum_{i=1}^{m} \log\big(\mathfrak{p}(y_i|\mathbf{x}_i;\mathbf{w})\big) \xrightarrow{\mathfrak{p}} m \cdot \mathbf{E}_{\mathfrak{p}(y|\mathbf{x};\mathbf{w}^*)}\Big[\log(\mathfrak{p}(y|\mathbf{x};\mathbf{w}))\Big] \\
&\qquad \implies -\arg\min_{\mathbf{w}} \mathbf{E}_{\mathfrak{p}(y|\mathbf{x};\mathbf{w}^*)}\left[\log\left(\frac{1}{\mathfrak{p}(y|\mathbf{x};\mathbf{w}))}\right)\right].
\end{aligned}$$

That is the empirical likelihood converges in probability to the expected population likelihood, and note going into the last line, minimizing the negative log-likelihood is what the optimizer solves. Now recall the KL-divergence expression of eqn (F.6), given current model settings $\mathfrak{p}(y|\mathbf{x};\mathbf{w})$, and optimal solution $\mathfrak{p}(y|\mathbf{x};\mathbf{w}^*)$, their divergence is:

$$\begin{aligned}
\arg\min_{\mathbf{w}} \mathsf{D}(\mathfrak{p}(y|\mathbf{x};\mathbf{w}^*)||\mathfrak{p}(y|\mathbf{x};\mathbf{w})) &= \mathbf{E}_{\mathfrak{p}(y|\mathbf{x};\mathbf{w}^*)}\left[\log\left(\frac{\mathfrak{p}(y|\mathbf{x};\mathbf{w}^*)}{\mathfrak{p}(y|\mathbf{x};\mathbf{w})}\right)\right] \\
&= \mathbf{E}_{\mathfrak{p}(y|\mathbf{x};\mathbf{w}^*)}\Big[\log \mathfrak{p}(y|\mathbf{x};\mathbf{w}^*) - \log \mathfrak{p}(y|\mathbf{x};\mathbf{w})\Big] \\
&\overset{(1)}{\sim} \mathbf{E}_{\mathfrak{p}(y|\mathbf{x};\mathbf{w}^*)}\Big[-\log \mathfrak{p}(y|\mathbf{x};\mathbf{w})\Big] \\
&= \mathbf{E}_{\mathfrak{p}(y|\mathbf{x};\mathbf{w}^*)}\left[\log\left(\frac{1}{\mathfrak{p}(y|\mathbf{x};\mathbf{w})}\right)\right].
\end{aligned}$$

Where going into (1), we use the fact that the term on left is a constant. So we see that maximizing the log-likelihood of data is the same as minimizing the KL-divergence between the optimal model parameter $\mathbf{w}^*$ and the current weights $\mathbf{w}$.

The $\boldsymbol{\sigma}(\cdot)$ function in kernalized logistic regression is an example of an *activation function*. Neural networks are built from repeated application of $\boldsymbol{\sigma}(\mathbf{w}^\intercal\mathbf{x})$ units. In fact, if we let the kth dimension of $\mathbf{w} \in \mathbb{R}^{k\times m}$ to go to infinity, then training this neural network with infinitely wide $\mathbf{w}$ is equivalent to training a two-layered logistic regression model.

E.4 Neural Tangent Kernel

This section continues the discussion of model selection in function space, and applies it to deep artificial neural networks (ANNs). In particular, we will show the equivalence between deep learning and kernel regression when the width of the intermediate layers go to infinity. This "Extra-wideness plays a crucial role in the proof: it is shown that as width increases, training causes increasingly smaller changes (in a proportionate sense) in the parameters. [AD+19]" This chapter will retrace many of these arguments:

1. the section "Neural Networks" will give a broad overview of ANNs, and their advantage over traditional kernel methods.
2. "Neural Network Gaussian Process" pose deep learning in two layered feed-forward networks as Gaussian process regression.
3. The last two sections will derive neural tangent kernels for multiplayer feed forward, and recurrent neural networks. Thus concluding the justification for ANN-derived kernel work units in the Metheus network.

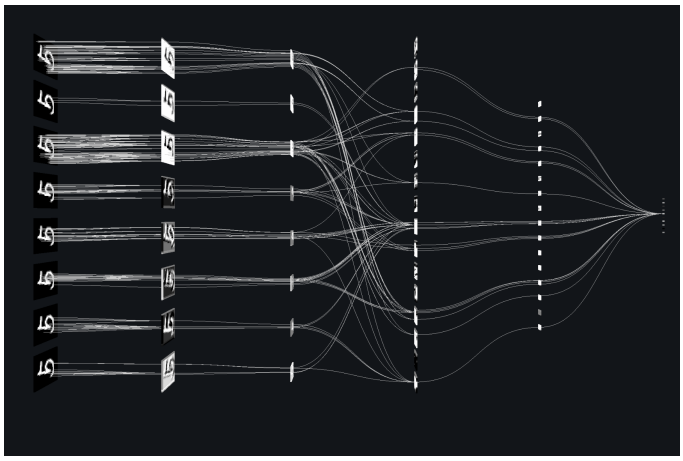


Figure E.7: Obligatory image of a neural network. This image is depicting MNIST classification using convolutional neural nets. Aside from that, it is unclear what else this picture is suppose to communicate. Most diagrams of neural networks are like this, albeit less artistic. This only deepens the mystique that the next few sections will dispel.

Neural Networks

Kernel based methods capture nonlinear relationship amongst input and output data, while maintaining the convexity of the loss function. However, there are some shortcomings when choosing the embedding function ϕ. One can either 1) opt for a very expressive infinite-order polynomial such as the Gaussian Kernel, or 2) manually engineer ϕ. The problem with the first approach is that overtly expressive kernels can over fit the data. In the latter approach, the statistician would need years of domain experience to construct a clever kernel. The deep learning approach is to learn the embedding alongside all model parameters, in other words learn the "basis functions" in an adaptive manner. For example in the kernel regression case of E.3.2, the model trained on m data points was of form:

$$y = f(\mathbf{x}; \mathbf{w}) = \langle \mathbf{w}, \phi_{\mathbf{x}} \rangle = \left\langle \sum_{\mathsf{i}=1}^{\mathsf{m}} \alpha_{\mathsf{i}} \kappa(\mathbf{x}_{\mathsf{i}}, \cdot),\ \kappa(\mathbf{x}, \cdot) \right\rangle = \left\langle \sum_{\mathsf{i}=1}^{\mathsf{m}} \alpha_{\mathsf{i}} \boldsymbol{\sigma}(\mathbf{B}\mathbf{x}_{\mathsf{i}}),\ \boldsymbol{\sigma}(\mathbf{B}\mathbf{x}) \right\rangle = \mathbf{A}\boldsymbol{\sigma}(\mathbf{B}\mathbf{x}).$$

where we use the expression $\boldsymbol{\sigma}(\mathbf{B}\mathbf{x}_{\mathsf{i}})$ to express the partially applied kernel function at the data point $\mathbf{x}_{\mathsf{i}}$. In the classical kernel case, $\mathbf{B}$ is some user defined block of polynomial functions. The activation function $\boldsymbol{\sigma}(\cdot)$ is some nonlinear function, i.e. e^x, whose parameters are not updated during learning. This $\boldsymbol{\sigma}(\cdot)$ serves as a decision function: i.e., it may average the signal going into a vertex on the computational graph, or it may take the maximum signal and squash its value so that it is between $[0, 1]$. In the ANN case, $\mathbf{B}$ is learned alongside the regression coefficients $\mathbf{A}$. That is $\mathbf{A}$ and $\mathbf{B}$ are functions of the data $(\mathbf{X}, \mathbf{y})$, and of each other. This interdependence renders the optimization problem non-convex. In the general L-layer-deep ANN, we put all the parameters in $\mathbf{w} = (\mathbf{A}, \mathbf{B}^1, \ldots, \mathbf{B}^L)$, and the model is:

$$f(\mathbf{x}; \mathbf{w}) = \mathbf{A}\boldsymbol{\sigma}(\mathbf{B}^L(\ldots \boldsymbol{\sigma}(\mathbf{B}^1(\mathbf{x}))\ldots)),$$

with L blocks of polynomials $\mathbf{B}^l$ transforming the input from the previous "layer," followed by nonlinear activation function $\boldsymbol{\sigma}(\cdot)$ [BMR21]. In order to learn the parameters, the optimizer repeats two tasks many times over for every lth layer:

1. construct a "function space" from "data" $\phi^{l-1}_{\mathbf{x}_{\mathsf{i}}}$ supplied by the $(l-1)$th layer. This space is spanned by the set of basis vectors $\{\phi^{l-1}_{\mathbf{x}_{\mathsf{i}}}\}$, where each basis vector is constructed with $\phi^{l-1}_{\mathbf{x}_{\mathsf{i}}} = \boldsymbol{\sigma}(\mathbf{B}^{l-1}\mathbf{x}^{l-1}_{\mathsf{i}})$.

2. Learn a set of polynomials $\mathbf{B}^l$ according to some decision criteria. This block will define a new function space, spanned by $\phi^l_{\mathbf{x}_\mathbf{i}} = \boldsymbol{\sigma}(\mathbf{B}^l\mathbf{x}^l_\mathbf{i})$ for the next layer $l+1$.

Each repetition of the above process constructs one layer in the deep ANN. At each layer, the "data" is supplied from the previous layer. So that at the bottom layer the "data" is just the $(\mathbf{X}, \mathbf{y})$ pairs sampled from nature, in the intermediate layer the data is output from models of the previous layer. Back-propagation makes end-to-end optimization of the model possible, so that every basis functions at each layer is learned together, along with the parameters of the final classification or regression objective parameterized by $\mathbf{A}$.

Now depending on the nature of data, its distribution can be summarized by two kinds of ANN architectures: feed-forward and recurrent neural nets.

1. The example shown in fig (E.7) is a *feed-forward networks*. They are used for data with global dependence, usually paired with global feature extractors such as convolution filters. Convolutional neural nets (CNNs) are the workhorse in this setting. In general, any data that can be "construed" as images with global information dependencies admit some solution with CNNs. This certainly applies to visual inputs of every kind, but it also find application in Alpha-Go, where the entire go-game-board is represented by the output from a CNN [SS+17].

2. *Recurrent neural networks (RNNs)* are nonlinear auto-regressive model used for time-series data. Dialogue systems with sentence generation is the canonical use case, trajectory generation for robotic arms moving through space is also a valid setting, as are path prediction for pedestrians crossing the road in a self-driving car application.

The last example is what makes deep ANNs compelling: the same architecture used for sentence parsing can also be used to evaluate pedestrians at a stop sign [[VS+17],[YBS22]]. Although the "depth" of deep ANNs capture the imagination of the public, most of the intuition for ANNs occur when the *width* of the intermediate layers are taken to infinity [Bel21]. In fact, it is known that arbitrarily wide ANN can approximate any function:

> "...standard multilayer feedforward networks with as few as one hidden layer using arbitrary squashing functions are capable of approximating any Borel measurable function from one finite dimensional space to another to any desired degree of accuracy, provided sufficiently many hidden units are available. In this sense, multilayer feedforward networks are a class of universal approximators [HSW89]."

Thus the logical point to start is to express a simple two-layered feed-forward neural networks with infinitely wide bottom layer. At initialization, this infinitely wide ANN is equivalent to Gaussian process regression, whereby the bottom layer forms a kernel, while the top layer perform linear summation of kernalized data [Nea96]. Furthermore:

> "...the evolution of an ANN during training can also be described by a kernel: ... the Neural Tangent Kernel (NTK). This kernel is central to describe the generalization features of ANNs. While the NTK is random at initialization and varies during training, in the infinite-width limit it converges to an explicit limiting kernel and it stays constant during training. This makes it possible to study the training of ANNs in function space instead of parameter space. Convergence of the training can then be related to the positive-definiteness of the limiting NTK [JG+18]."

Most saliently, the training of ANN follows a descent along the kernel gradient w.r.t the NTK, which is convex. *That is to say that training some ANNs under specific conditions is equivalent to (NTK) kernel regression.* This analysis generalize to feed-forward networks of arbitrary depth, and recurrent neural networks as well. In practice ANNs do not have

infinite width, and yet , there is "excellent empirical agreement between the predictions of [infinitely wide ANNs] and those of the linearized version even for finite practically-sized networks" [LX+19]. The next three sections expounds on this point. The arguments are drawn from [JG+18] and [LZB21], but follows the gentler approach of [Rad+22b], [Rad+22d], and [Rad+22c].

Feed-forward Neural Networks as Gaussian Process

(Setting) Similar to least squares shown in example E.3.2, the data is assumed to be generated by $\mathsf{y} = f(\mathbf{x}; \mathbf{w}) = \langle \mathbf{w}, \mathbf{x} \rangle + \epsilon$, where $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. In the ANN setting, we gather all the parameters in $\mathbf{w} = (\mathbf{A}, \mathbf{B})$, and define the network $f(\mathbf{x}; \mathbf{w})$ with:

$$
\begin{aligned}
& f : \mathbb{R}^{\mathsf{d}_0} \to \mathbb{R} \\
& f(\mathbf{x}; \mathbf{w}) = \frac{1}{\sqrt{\mathsf{d}_1}} \mathbf{A} \boldsymbol{\sigma}(\mathbf{B}\mathbf{x}) \qquad \text{for } \mathbf{y} \in \mathbb{R} \;\; \mathbf{x} \in \mathbb{R}^{\mathsf{d}_0} \qquad \text{with } \mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \\
& \text{where } \mathbf{A} \in \mathbb{R}^{\mathbf{1} \times \mathsf{d}_1} \qquad \mathbf{B} \in \mathbb{R}^{\mathsf{d}_1 \times \mathsf{d}_0} \qquad \boldsymbol{\sigma}(\cdot) : \mathbb{R} \to \mathbb{R}.
\end{aligned} \tag{E.16}
$$

Some clarifications:

1. The activation function $\boldsymbol{\sigma}(\cdot)$ is applied element wise.
2. The $\frac{1}{\sqrt{\mathsf{d}_1}}$ is a numerical convenience that we will explain later. Note we can always just multiply the final values of $\mathbf{A}$ by $1/\sqrt{\mathsf{d}_1}$, and thus remove this constant term from the expression
3. The vectors $\mathbf{x}$ are normalized, and are drawn i.i.d from the surface of a sphere with dimension $\mathsf{d}_0 - 1$, see fig (E.8). In practice this is not a stringent assumption as the data can always be zero-meaned and normalized to be between 0 and 1.

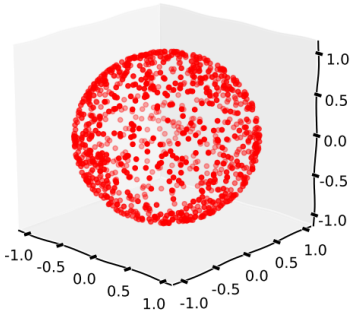


Figure E.8: The data are drawn $\mathbf{x} \overset{\text{iid}}{\sim} \mathcal{N}(\mathbf{0}, \mathbf{I})$, where $\mathbf{0} \in \mathbb{R}^{\mathsf{d}_0 - 1}$, and $\mathbf{I} \in \mathbb{R}^{\mathsf{d}_0 - 1 \times \mathsf{d}_0 - 1}$. That is the sphere is lying in ambient space with dimension d_0, but the distribution has support in $\mathsf{d}_0 - 1$. This is a very common assumption since for high values of d_0, most of the volume of the sphere are on the surface, not the interior. So a randomly generated set of vectors will be on the surface.

Now given data set:

$$(\mathbf{X} \in \mathbb{R}^{\mathsf{m} \times \mathsf{d}_0}, \;\; \mathbf{y} \in \mathbb{R}^{\mathsf{m} \times 1}),$$

the model is trained by minimizing the least squares error using gradient descent:

$$\mathsf{loss}(\mathbf{A}, \mathbf{B}) = \sum_{\mathsf{i}=1}^{\mathsf{m}} (\mathsf{y}_\mathsf{i} - f(\mathbf{x}_\mathsf{i}))^2 = \sum_{\mathsf{i}=1}^{\mathsf{m}} (\mathsf{y}_\mathsf{i} - \mathbf{A}\boldsymbol{\sigma}(\mathbf{B}\mathbf{x}_\mathsf{i}))^2. \tag{E.17}$$

Note this objective is not convex as $\mathbf{A}$ and $\mathbf{B}$ are both optimized. That is given f initialized at $(\mathbf{A}^o, \hat{\mathbf{B}})$, the two matrices are updated in parallel via gradient descent with:

$$
\begin{aligned}
\mathbf{A}^{t+1} &\leftarrow \mathbf{A}^{\mathsf{t}} - \eta \nabla \mathsf{loss}_{\mathbf{A}}(\mathbf{A}^t, \mathbf{B}^t) \\
\mathbf{B}^{t+1} &\leftarrow \mathbf{B}^{\mathsf{t}} - \eta \nabla \mathsf{loss}_{\mathbf{B}}(\mathbf{A}^t, \mathbf{B}^t).
\end{aligned}
$$

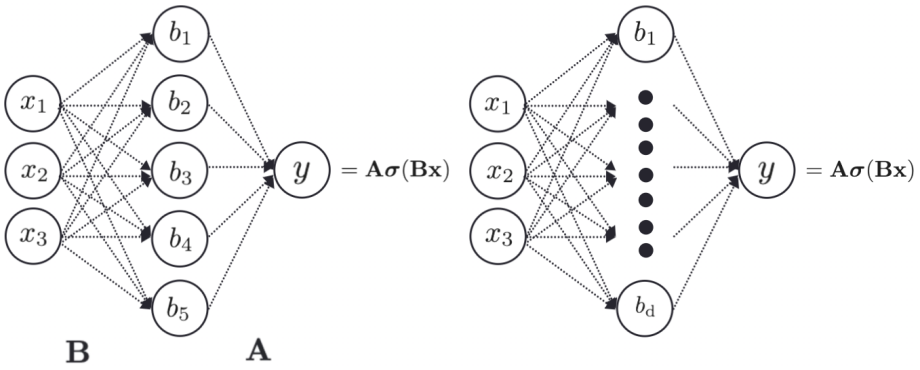


Figure E.9: A two layer neural network $\mathsf{y} = \mathbf{A}\boldsymbol{\sigma}(\mathbf{B}\mathbf{x})$, where $\mathbf{x} = (x_1, x_2, x_3) \in \mathbb{R}^3$. Left: the feature matrix where $\mathrm{d}_1 = 5$, so it has dimension $\mathbf{B} \in \mathbb{R}^{3\times 5}$, thus mapping $\mathbf{x}$ to some vector in $\mathbb{R}^5$. The output matrix has dimension $\mathbf{A} \in \mathbb{R}^{5\times 1}$, and maps the inputs $\mathbf{x}$ to some label y. Right: in the limit, the d_1 dimension of $\mathbf{B} \in \mathbb{R}^{3\times \mathrm{d}_1}$ grows to infinity. Now consider a setup where the entries of the infinite dimensional $\mathbf{B}$ are fixed by sampling via $\mathbf{B}_{\mathsf{ij}} \sim \mathcal{N}(0, 1)$. Given fixed $\hat{\mathbf{B}}$, if we just train $\mathbf{A}$ using least squares loss, then this setting is the same as kernel regression, where $\boldsymbol{\sigma}(\hat{\mathbf{B}}\mathbf{x})$ is the kernel. This can be generalized to neural nets of arbitrary depth, so that training the kth layer of a neural network, whilst fixing the bottom $k-1$ layers, is just regression. Kernel regression can be posed in a non-parametric manner using the tools of Gaussian process, this motivates the derivation of neural tangent kernels for different activation functions [Rad+22b].

The next proposition will recover convexity by simplifying the problem. It states that if the values of the feature matrix $\mathbf{B}$ are randomly initialized and then frozen, then eqn (E.17) reduces to kernel regression.

Proposition E.4.1. *(Neural Network Gaussian Process - NNGP) Given the setting above, suppose the learner initializes the values of* $\mathbf{B}$ *by sampling its entries i.i.d from* $\mathcal{N}(\mathbf{0}, \mathbf{I})$*, and then freezes the weights of this* $\hat{\mathbf{B}}$*. Then training with the loss by varying* $\mathbf{A}$ *while fixing* $\hat{\mathbf{B}}$*:*

$$\mathsf{loss}(\mathbf{A}) = \sum_{i=1}^{m} (\mathsf{y}_i - \mathbf{A}\boldsymbol{\sigma}(\hat{\mathbf{B}}\mathbf{x}_i))^2, \tag{E.18}$$

is equivalent to kernalized linear regression. Here the basis functions of the feature space $\mathcal{H}_\kappa$ *is defined by* $\phi_{\mathbf{x}_i} = \boldsymbol{\sigma}(\hat{\mathbf{B}}\mathbf{x}_i)$*, for every* $\mathbf{x}_i$ *in the training set. In particular, as the width of* $\hat{\mathbf{B}}$ *approaches infinity, that is* $d_1 \to \infty$*, we have:*

E.4.1.1. *the solution to regression is given by:*

$$f(\mathbf{x}; \mathbf{w}) \sim \sum_{i=1}^{d_1} \alpha_i \boldsymbol{\sigma}(\langle \hat{\mathbf{b}}_i, \mathbf{x} \rangle). \tag{E.19}$$

E.4.1.2. *The kernel* $\kappa(\mathbf{x}_i, \mathbf{x}_j)$ *is defined as:*

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \left\langle \frac{1}{\sqrt{d_1}} \phi_{\mathbf{x}_i}, \frac{1}{\sqrt{d_1}} \phi_{\mathbf{x}_j} \right\rangle. \tag{E.20}$$

Moreover, the value of this kernel is a r.v., by definition. The source of randomness are the data drawn by $\mathbf{x}_i, \mathbf{x}_j \overset{\mathsf{iid}}{\sim} \mathcal{N}(\mathbf{0}, \mathbf{I})$*, and entries of* $\mathbf{B}$ *populated according to* $(\hat{\mathbf{B}})_{ij} \overset{\mathsf{iid}}{\sim} \mathcal{N}(0, 1)$*.*

Thus the kernel values are drawn according to $\kappa(\mathbf{x}_i, \mathbf{x}_j) \overset{\text{iid}}{\sim} \mathcal{N}(0, \Lambda)$*: centered at zero, with covariance:*

$$\Lambda = \begin{bmatrix} \langle \mathbf{x}_i, \mathbf{x}_i \rangle & \langle \mathbf{x}_i, \mathbf{x}_j \rangle \\ \langle \mathbf{x}_j, \mathbf{x}_i \rangle & \langle \mathbf{x}_j, \mathbf{x}_j \rangle \end{bmatrix}.$$

A diagram of the setting is provided in fig (E.9).

Proof. The expected value of the function $\mathbf{E}[f(\mathbf{x}; \mathbf{w})]$ is a straight forward derivation. We can write the terms of $\mathbf{A}$ and $\hat{\mathbf{B}}$ as:

$$\mathbf{A} = (\alpha_1, \ldots, \alpha_{\mathrm{d}_1}) \qquad \hat{\mathbf{B}} = \begin{bmatrix} \hat{b}_{11} & \ldots & \hat{b}_{1\mathrm{d}_0} \\ \vdots & \ddots & \vdots \\ \hat{b}_{\mathrm{d}_1 1} & \ldots & \hat{b}_{\mathrm{d}_1 \mathrm{d}_0} \end{bmatrix} = \begin{bmatrix} — \hat{\mathbf{b}}_1 — \\ \vdots \\ — \hat{\mathbf{b}}_{\mathrm{d}_1} — \end{bmatrix},$$

where $\mathbf{b}_\mathsf{r} \in \mathbb{R}^{1 \times \mathrm{d}_0}$ are the rth row vectors for $r = 1, \ldots, \mathrm{d}_1$. Then for some $\mathbf{x} = (x_1, \ldots, x_{\mathrm{d}_0})$, we can rewrite the ANN with:

$$\begin{aligned} f(\mathbf{x}; \mathbf{w}) &= \frac{1}{\sqrt{\mathrm{d}_1}} \mathbf{A} \boldsymbol{\sigma}(\hat{\mathbf{B}} \mathbf{x}) \\ &= \frac{1}{\sqrt{\mathrm{d}_1}} \mathbf{A} \boldsymbol{\sigma}\left(\begin{bmatrix} —— \hat{\mathbf{b}}_1 —— \\ \vdots \\ —— \hat{\mathbf{b}}_{\mathrm{d}_1} —— \end{bmatrix} \mathbf{x}^\intercal \right) = \frac{1}{\sqrt{\mathrm{d}_1}} (\alpha_1, \ldots, \alpha_{\mathrm{d}_1}) \cdot \begin{bmatrix} \boldsymbol{\sigma}(\langle \hat{\mathbf{b}}_1, \mathbf{x} \rangle) \\ \vdots \\ \boldsymbol{\sigma}(\langle \hat{\mathbf{b}}_{\mathrm{d}_1}, \mathbf{x} \rangle) \end{bmatrix} \\ &= \frac{1}{\sqrt{\mathrm{d}_1}} \sum_{\mathsf{r}=1}^{\mathrm{d}_1} \alpha_\mathsf{r} \boldsymbol{\sigma}(\langle \hat{\mathbf{b}}_\mathsf{r}, \mathbf{x} \rangle). \end{aligned} \tag{E.21}$$

Some observations:

1. The model $f(\mathbf{x}; \mathbf{w})$ is a *linear* summation of the featurized data, this is a consequence of fixing the feature matrix to $\hat{\mathbf{B}}$.
2. The value $\mathbf{x}$ is featurized with the function $\boldsymbol{\sigma}(\langle \hat{\mathbf{b}}_\mathsf{r}, \mathbf{x} \rangle)$, with randomly initialized $\hat{\mathbf{b}}_r$. If the $\mathbf{b}_r$ is constructed from the training set instead, then we would recover the kernel expression of eqn (E.3.2).
3. The sum is over d_1, so that as $\mathrm{d}_1 \to \infty$, $\mathbf{E}[f(\mathbf{x}; \mathbf{w})]$ is Gaussian by definition, see fig (**??**).

Now given $\mathbf{x}_\mathsf{i}, \mathbf{x}_\mathsf{j} \in \mathbb{R}^{\mathrm{d}_0}$, we use the definition of the kernel from eqn (E.10) to write its expected value as $\mathrm{d}_1 \to \infty$:

$$\begin{aligned} \kappa(\mathbf{x}_\mathsf{i}, \mathbf{x}_\mathsf{j}) &= \left\langle \frac{1}{\sqrt{\mathrm{d}_1}} \phi_{\mathbf{x}_\mathsf{i}}, \frac{1}{\sqrt{\mathrm{d}_1}} \phi_{\mathbf{x}_\mathsf{j}} \right\rangle \\ &= \left\langle \frac{1}{\sqrt{\mathrm{d}_1}} \boldsymbol{\sigma}(\hat{\mathbf{B}} \mathbf{x}_\mathsf{i}), \frac{1}{\sqrt{\mathrm{d}_1}} \boldsymbol{\sigma}(\hat{\mathbf{B}} \mathbf{x}_\mathsf{j}) \right\rangle \\ &= \frac{1}{\sqrt{\mathrm{d}_1}} \boldsymbol{\sigma}\left(\begin{bmatrix} —— \hat{\mathbf{b}}_1 —— \\ \vdots \\ —— \hat{\mathbf{b}}_{\mathrm{d}_1} —— \end{bmatrix} \mathbf{x}_\mathsf{i}^\intercal \right)^\intercal \cdot \frac{1}{\sqrt{\mathrm{d}_1}} \boldsymbol{\sigma}\left(\begin{bmatrix} —— \hat{\mathbf{b}}_1 —— \\ \vdots \\ —— \hat{\mathbf{b}}_{\mathrm{d}_1} —— \end{bmatrix} \mathbf{x}_\mathsf{j}^\intercal \right) \\ &= \frac{1}{\mathrm{d}_1} \left[\boldsymbol{\sigma}(\langle \hat{\mathbf{b}}_1, \mathbf{x}_\mathsf{i} \rangle), \ldots, \boldsymbol{\sigma}(\langle \hat{\mathbf{b}}_{\mathrm{d}_1}, \mathbf{x}_\mathsf{i} \rangle) \right] \cdot \begin{bmatrix} \boldsymbol{\sigma}(\langle \hat{\mathbf{b}}_1, \mathbf{x}_\mathsf{j} \rangle) \\ \vdots \\ \boldsymbol{\sigma}(\langle \hat{\mathbf{b}}_{\mathrm{d}_1}, \mathbf{x}_\mathsf{j} \rangle) \end{bmatrix} \end{aligned}$$

$$= \frac{1}{\mathrm{d}_1}\sum_{\mathsf{r}=1}^{\mathrm{d}_1} \boldsymbol{\sigma}(\langle \hat{\mathbf{b}}_\mathsf{r}, \mathbf{x}_\mathsf{i}\rangle)\cdot \boldsymbol{\sigma}(\langle \hat{\mathbf{b}}_\mathsf{r}, \mathbf{x}_\mathsf{j}\rangle)$$
$$\overset{(1)}{=} \mathbf{E}_{\hat{\mathbf{B}}\sim\mathcal{N}(0,\mathbf{I})}\left[\left\langle \boldsymbol{\sigma}(\hat{\mathbf{B}}\mathbf{x}_\mathsf{i}),\ \boldsymbol{\sigma}(\hat{\mathbf{B}}\mathbf{x}_\mathsf{j})\right\rangle\right]. \tag{E.22}$$

Whereby going into (1), we take the limit $\mathrm{d}_1 \to \infty$ and use the law of large numbers to rewrite the sum as converging to the expectation of the inner product in probability. That is:

$$\lim_{\mathrm{d}_1\to\infty}\frac{1}{\mathrm{d}_1}\sum_{\mathsf{r}=1}^{\mathrm{d}_1} \boldsymbol{\sigma}(\langle \hat{\mathbf{b}}_\mathsf{r}, \mathbf{x}_\mathsf{i}\rangle)\cdot \boldsymbol{\sigma}(\langle \hat{\mathbf{b}}_\mathsf{r}, \mathbf{x}_\mathsf{j}\rangle) \ \xrightarrow{\mathsf{p}}\ \mathbf{E}_{\hat{\mathbf{b}}_\mathsf{r}\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\left[\left\langle \boldsymbol{\sigma}(\hat{\mathbf{B}}\mathbf{x}_\mathsf{i}),\ \boldsymbol{\sigma}(\hat{\mathbf{B}}\mathbf{x}_\mathsf{j})\right\rangle\right].$$

Now first observe the expected value of $\kappa(\mathbf{x}_\mathsf{i}, \mathbf{x}_\mathsf{j})$ is zero, since the rcth entry of feature matrix is drawn with $\hat{\mathbf{B}}_{\mathsf{rc}} \overset{\mathsf{iid}}{\sim} \mathcal{N}(0,1)$, and $\mathbf{x}_\mathsf{i} \overset{\mathsf{iid}}{\sim} \mathcal{N}(\mathbf{0},\mathbf{I})$ by assumption. So the product of two centered Gaussian is another centered Gaussian. Next note the expectation is taken w.r.t the random variables $\hat{\mathbf{B}}_{\mathsf{rc}}$, however we want to write the value of the kernel $\kappa(\mathbf{x}_\mathsf{i}, \mathbf{x}_\mathsf{j})$ as drawn from the joint distribution over the pair of random variables $(\langle \hat{\mathbf{b}}_\mathsf{r}, \mathbf{x}_\mathsf{i}\rangle, \langle \hat{\mathbf{b}}_\mathsf{r}, \mathbf{x}_\mathsf{j}\rangle)$:

$$\mathbf{E}_{(\langle \hat{\mathbf{b}}_\mathsf{r},\mathbf{x}_\mathsf{i}\rangle, \langle \hat{\mathbf{b}}_\mathsf{r},\mathbf{x}_\mathsf{j}\rangle)\sim\mathcal{N}(\mathbf{0},\Lambda)}\left[\left\langle \boldsymbol{\sigma}(\hat{\mathbf{B}}\mathbf{x}_\mathsf{i}),\ \boldsymbol{\sigma}(\hat{\mathbf{B}}\mathbf{x}_\mathsf{j})\right\rangle\right]$$
$$\Lambda = \begin{bmatrix} cov(\langle \hat{\mathbf{b}}_\mathsf{r}, \mathbf{x}_\mathsf{i}\rangle, \langle \hat{\mathbf{b}}_\mathsf{r}, \mathbf{x}_\mathsf{i}\rangle) & cov(\langle \hat{\mathbf{b}}_\mathsf{r}, \mathbf{x}_\mathsf{i}\rangle, \langle \hat{\mathbf{b}}_\mathsf{r}, \mathbf{x}_\mathsf{j}\rangle) \\ cov(\langle \hat{\mathbf{b}}_\mathsf{r}, \mathbf{x}_\mathsf{j}\rangle, \langle \hat{\mathbf{b}}_\mathsf{r}, \mathbf{x}_\mathsf{i}\rangle) & cov(\langle \hat{\mathbf{b}}_\mathsf{r}, \mathbf{x}_\mathsf{j}\rangle, \langle \hat{\mathbf{b}}_\mathsf{r}, \mathbf{x}_\mathsf{j}\rangle) \end{bmatrix}.$$

The mean is once again $\mathbf{E}[\langle \hat{\mathbf{b}}_\mathsf{r}, \mathbf{x}_\mathsf{i}\rangle] = \mathbf{E}[\langle \hat{\mathbf{b}}_\mathsf{r}, \mathbf{x}_\mathsf{j}\rangle] = 0$. Now consider their covariance, fixing two constants $\mathbf{x}_\mathsf{i}, \mathbf{x}_\mathsf{j}$ we can compute their covariance with:

$$cov\big(\langle \hat{\mathbf{b}}_\mathsf{r}, \mathbf{x}_\mathsf{i}\rangle, \langle \hat{\mathbf{b}}_\mathsf{r}, \mathbf{x}_\mathsf{j}\rangle\big) = \mathbf{E}_{\hat{\mathbf{b}}_\mathsf{r}\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\left[\langle \hat{\mathbf{b}}_\mathsf{r}, \mathbf{x}_\mathsf{i}\rangle - \mathbf{E}[\langle \hat{\mathbf{b}}_\mathsf{r}, \mathbf{x}_\mathsf{i}\rangle]\right]\cdot \mathbf{E}_{\hat{\mathbf{b}}_\mathsf{r}\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\left[\langle \hat{\mathbf{b}}_\mathsf{r}, \mathbf{x}_\mathsf{j}\rangle - \mathbf{E}[\langle \hat{\mathbf{b}}_\mathsf{r}, \mathbf{x}_\mathsf{j}\rangle]\right]$$
$$\overset{(1)}{=} \mathbf{E}_{\hat{\mathbf{b}}_\mathsf{r}}\left[\langle \hat{\mathbf{b}}_\mathsf{r}, \mathbf{x}_\mathsf{i}\rangle\right]\cdot \mathbf{E}_{\hat{\mathbf{b}}_\mathsf{r}}\left[\langle \hat{\mathbf{b}}_\mathsf{r}, \mathbf{x}_\mathsf{j}\rangle\right] \overset{(2)}{=} \mathbf{E}_{\hat{\mathbf{b}}_\mathsf{r}}\left[\sum_{\mathsf{c}=1}^{\mathrm{d}_0}\hat{b}_{\mathsf{rc}}x_{\mathsf{ic}}\right]\cdot \mathbf{E}_{\hat{\mathbf{b}}_\mathsf{r}}\left[\sum_{\mathsf{c}=1}^{\mathrm{d}_0}\hat{b}_{\mathsf{rc}}x_{\mathsf{jc}}\right]$$
$$\overset{(3)}{=} \sum_{\mathsf{c}=1}^{\mathrm{d}_0} x_{\mathsf{ic}}\cdot \mathbf{E}_{\hat{\mathbf{b}}_\mathsf{r}}[\hat{b}_{\mathsf{rc}}]\cdot \sum_{\mathsf{c}=1}^{\mathrm{d}_0} x_{\mathsf{jc}}\cdot \mathbf{E}_{\hat{\mathbf{b}}_\mathsf{r}}[\hat{b}_{\mathsf{rc}}] \overset{(4)}{=} \sum_{\mathsf{c}=1}^{\mathrm{d}_0} x_{\mathsf{ic}}\cdot x_{\mathsf{jc}}\cdot (\mathbf{E}_{\hat{\mathbf{b}}_\mathsf{r}}[\hat{b}_{\mathsf{rc}}])^2$$
$$\overset{(5)}{=} \mathbf{E}_{\hat{\mathbf{b}}_\mathsf{r}}\left[\sum_{\mathsf{c}=1}^{\mathrm{d}_0} x_{\mathsf{ic}}\cdot x_{\mathsf{jc}}\cdot \hat{b}_{\mathsf{rc}}^2\right] \overset{(6)}{=} \langle \mathbf{x}_\mathsf{i}, \mathbf{x}_\mathsf{j}\rangle.$$

Where going into (1), we use the fact that $\mathbf{E}_{\hat{\mathbf{b}}_\mathsf{r}\sim\mathcal{N}(0,\mathbf{I})}[\langle \hat{\mathbf{b}}_\mathsf{r}, \mathbf{x}_\mathsf{i}\rangle] = 0$ for every $\mathbf{x}_\mathsf{i}$. Going into (2) uses the definition of inner product by writing $\mathbf{x}_\mathsf{i} = (x_{i1}, .., x_{\mathsf{ic}}, .., x_{i\mathrm{d}_0})$, and $\hat{\mathbf{b}}_r = (\hat{b}_{r1}, .., \hat{b}_{\mathsf{rc}}, .., \hat{b}_{r\mathrm{d}_0})$. (3) is by linearity of expectation, and the fact that $x_{\mathsf{ic}}, x_{\mathsf{jc}}$ does not depend on the expectation. Going into (4) we use the fact that the cross product terms $\sum_{c,d=1}^{\mathrm{d}_0} x_{\mathsf{ic}}x_{jd}\mathbf{E}[\hat{b}_{\mathsf{rc}}]\mathbf{E}[\hat{b}_{rd}] = 0$, since the b_{rc} and b_{rd} terms are independent. In (5) we use the linearity of expectations and the fact that $\mathbf{E}_{\hat{\mathbf{b}}_\mathsf{r}}[x_{\mathsf{ic}}] = x_{\mathsf{ic}}$ for $\mathbf{x}_\mathsf{i}$ and $\mathbf{x}_\mathsf{j}$. In (6) we use the definition of the inner product to rewrite the sum. A similar argument holds for covariance over $cov(\langle \hat{\mathbf{b}}_\mathsf{r}, \mathbf{x}_\mathsf{j}\rangle, \langle \hat{\mathbf{b}}_\mathsf{r}, \mathbf{x}_\mathsf{j}\rangle)$, etc. Thus we have:

$$\kappa(\mathbf{x}_\mathsf{i}, \mathbf{x}_\mathsf{j}) = \mathbf{E}_{(\langle \hat{\mathbf{b}}_\mathsf{r},\mathbf{x}_\mathsf{i}\rangle, \langle \hat{\mathbf{b}}_\mathsf{r},\mathbf{x}_\mathsf{j}\rangle)\sim\mathcal{N}(\mathbf{0},\Lambda)}\left[\left\langle \boldsymbol{\sigma}(\mathbf{B}\mathbf{x}_\mathsf{i}),\ \boldsymbol{\sigma}(\mathbf{B}\mathbf{x}_\mathsf{j})\right\rangle\right]$$
$$\Lambda = \begin{bmatrix} \langle \mathbf{x}_\mathsf{i}, \mathbf{x}_\mathsf{i}\rangle & \langle \mathbf{x}_\mathsf{i}, \mathbf{x}_\mathsf{j}\rangle \\ \langle \mathbf{x}_\mathsf{j}, \mathbf{x}_\mathsf{i}\rangle & \langle \mathbf{x}_\mathsf{j}, \mathbf{x}_\mathsf{j}\rangle \end{bmatrix}, \tag{E.23}$$

with $\phi_{\mathbf{x}_\mathsf{i}} = \boldsymbol{\sigma}(\mathbf{B}\mathbf{x}_\mathsf{i})$ as desired [Rad+22b]. ■

(Activation Functions) Now we will find an explicit expression for the expectation of eqn ($E.23$) for specific activation functions.

1. As a warm up, consider the activation $\boldsymbol{\sigma}(x) = e^{ix}$, for some imaginary number i and $x \in \mathbb{R}$. The inner product is defined as:

$$\Big\langle \{a_n\}_{n\geq 0},\ \{b_n\}_{n\geq 0} \Big\rangle = \sum_{n\geq 0} a_n \overline{b_n}.$$

   Thus an ANN using e^{ix} as the activation function has kernel:

$$\begin{aligned}
\kappa(\mathbf{x}_\mathsf{i}, \mathbf{x}_\mathsf{j}) &= \mathbf{E}_{\langle \mathbf{b}_\mathsf{r}, \mathbf{x}_\mathsf{i}\rangle, \langle \mathbf{b}_\mathsf{r}, \mathbf{x}_\mathsf{j}\rangle \sim \mathcal{N}(\mathbf{0}, \Lambda)} \left[\Big\langle \boldsymbol{\sigma}(\mathbf{B}\mathbf{x}_\mathsf{i}),\ \overline{\boldsymbol{\sigma}(\mathbf{B}\mathbf{x}_\mathsf{j})} \Big\rangle \right] \\
&= \mathbf{E}_{\langle \hat{\mathbf{b}}_\mathsf{r}, \mathbf{x}_\mathsf{i}\rangle, \langle \mathbf{b}_\mathsf{r}, \mathbf{x}_\mathsf{j}\rangle \sim \mathcal{N}(\mathbf{0}, \Lambda)} \left[\exp\Big(i \cdot (\langle \mathbf{b}_\mathsf{r}, \mathbf{x}_\mathsf{i}\rangle - \langle \mathbf{b}_\mathsf{r}, \mathbf{x}_\mathsf{j}\rangle) \Big) \right] \\
&= \prod_{\mathsf{c}=1}^{\mathsf{d}_0} \mathbf{E}_{\hat{b}_{\mathsf{rc}} \sim \mathcal{N}(0,1)} \left[\exp\Big(i b_{\mathsf{rc}} \cdot (x_{\mathsf{ic}} - x_{\mathsf{jc}}) \Big) \right] \\
&= \prod_{\mathsf{c}=1}^{\mathsf{d}_0} \frac{1}{2\pi} \int_{-\infty}^{\infty} \exp\Big(i b_{\mathsf{rc}} \cdot (x_{\mathsf{ic}} - x_{\mathsf{jc}}) \Big) \exp\Big(\frac{b_{\mathsf{rc}}^2}{2} \Big) d\nu(b_{\mathsf{rc}}) \\
&= \prod_{\mathsf{c}=1}^{\mathsf{d}_0} \exp\Big(-\frac{(x_{\mathsf{ic}} - x_{\mathsf{jc}})^2}{2} \Big) \frac{1}{2\pi} \int_{-\infty}^{\infty} \exp\Big(\frac{(b_{\mathsf{rc}} - i(x_{\mathsf{ic}} - x_{\mathsf{jc}}))^2}{2} \Big) d\nu(b_{\mathsf{rc}}) \\
&= \prod_{\mathsf{c}=1}^{\mathsf{d}_0} \exp\Big(-\frac{(x_{\mathsf{ic}} - x_{\mathsf{jc}})^2}{2} \Big) = \exp\Big(-\frac{1}{2} \|\mathbf{x}_\mathsf{i} - \mathbf{x}_\mathsf{j}\|_2^2 \Big).
\end{aligned}$$

   This is just the Gaussian kernel where the length parameter is set to 1 [Rad+22b].

2. *ReLU activation*: this is the most common activation today, defined with:

$$\mathrm{ReLU}(x) = \boldsymbol{\sigma}(x) = max(0, x). \tag{E.24}$$

   Applying the definition of expectation, the kernel value becomes:

$$\begin{aligned}
\kappa(\mathbf{x}_\mathsf{i}, \mathbf{x}_\mathsf{j}) &= \mathbf{E}_{\langle \hat{\mathbf{b}}_\mathsf{r}, \mathbf{x}_\mathsf{i}\rangle, \langle \hat{\mathbf{b}}_\mathsf{r}, \mathbf{x}_\mathsf{j}\rangle \sim \mathcal{N}(\mathbf{0}, \Lambda)} \left[\Big\langle \boldsymbol{\sigma}(\mathbf{B}\mathbf{x}_\mathsf{i}),\ \boldsymbol{\sigma}(\mathbf{B}\mathbf{x}_\mathsf{j}) \Big\rangle \right] \\
&= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \Big\langle \boldsymbol{\sigma}(\mathbf{B}\mathbf{x}_\mathsf{i}),\ \boldsymbol{\sigma}(\mathbf{B}\mathbf{x}_\mathsf{j}) \Big\rangle \mathfrak{p}\Big(\langle \hat{\mathbf{b}}_\mathsf{r}, \mathbf{x}_\mathsf{i}\rangle, \langle \hat{\mathbf{b}}_\mathsf{r}, \mathbf{x}_\mathsf{j}\rangle : \mathbf{0}, \Lambda \Big) d(\langle \hat{\mathbf{b}}_\mathsf{r}, \mathbf{x}_\mathsf{i}\rangle) d(\langle \hat{\mathbf{b}}_\mathsf{r}, \mathbf{x}_\mathsf{j}\rangle),
\end{aligned}$$

   where the $\mathfrak{p}(\cdot)$ is the Gaussian distribution of eqn (**??**). The complete calculation can be found in [Rad+22b]. The expectation of the kernel reduces to:

$$\kappa(\mathbf{x}_\mathsf{i}, \mathbf{x}_\mathsf{j}) = \frac{1}{\pi} \Big(\langle \mathbf{x}_\mathsf{i}, \mathbf{x}_\mathsf{j}\rangle \Big(\pi - cos^{-1} \frac{\langle \mathbf{x}_\mathsf{i}, \mathbf{x}_\mathsf{j}\rangle}{\|\mathbf{x}_\mathsf{i}\|_2 \|\mathbf{x}_\mathsf{j}\|} \Big) + \sqrt{\|\mathbf{x}_\mathsf{i}\|_2^2 \|\mathbf{x}_\mathsf{j}\|_2^2 - \langle \mathbf{x}_\mathsf{i}, \mathbf{x}_\mathsf{j}\rangle^2} \Big), \tag{E.25}$$

   where $\|\mathbf{x}\|_2^2 = \langle \mathbf{x}, \mathbf{x}\rangle$. This is a periodic kernel with some variable shift. And note the values of the feature matrix $\mathbf{B}$ do not appear because we fixed $\hat{\mathbf{B}}$, so that $cov(\mathbf{x}_\mathsf{i}, \mathbf{x}_\mathsf{j}) = \langle \mathbf{x}_\mathsf{i}, \mathbf{x}_\mathsf{j}\rangle$.

3. *Dual ReLU activation*: recall in our case the data points lie on the surface of the unit sphere, so that their squared norm is 1: $\|\mathbf{x}\|_2^2 = \langle \mathbf{x}, \mathbf{x}\rangle = 1$. One can define a dual activation function $\ddot{\boldsymbol{\sigma}}(\cdot) : [-1, 1] \to \mathbb{R}$, so that eqn (E.23) above simplifies to:

$$\begin{aligned}
\kappa(\mathbf{x}_\mathsf{i}, \mathbf{x}_\mathsf{j}) &= \mathbf{E}_{\langle \hat{\mathbf{b}}_\mathsf{r}, \mathbf{x}_\mathsf{i}\rangle, \langle \hat{\mathbf{b}}_\mathsf{r}, \mathbf{x}_\mathsf{j}\rangle \sim \mathcal{N}(\mathbf{0}, \Lambda)} \left[\Big\langle \ddot{\boldsymbol{\sigma}}(\mathbf{B}\mathbf{x}_\mathsf{i}),\ \ddot{\boldsymbol{\sigma}}(\mathbf{B}\mathbf{x}_\mathsf{j}) \Big\rangle \right] \overset{(1)}{=} \ddot{\boldsymbol{\sigma}}(\mathbf{x}_\mathsf{i}^\intercal \mathbf{x}_\mathsf{j}) \\
\Lambda &= \begin{bmatrix} 1 & \langle \mathbf{x}_\mathsf{i}, \mathbf{x}_\mathsf{j}\rangle \\ \langle \mathbf{x}_\mathsf{j}, \mathbf{x}_\mathsf{i}\rangle & 1 \end{bmatrix}.
\end{aligned} \tag{E.26}$$

Where going into (1), we exchange the application the dual activation with the inner product with the following identity.

Proposition E.4.2. *(Hermite Polynomials) Define the Hermite polynomials* $\{h_i\}_{i\geq 0}$ *as the orthonormal basis for the Hilbert space* $L^2(\cdot)$ *(see eqn (??)), with inner product* $\langle f, g\rangle = \int_{\mathbb{R}} f(x)g(x)e^{-x^2/2}dx$*, constructed by performing Gram-Schmidt orthogonalization on* $L^2(\cdot)$ *using the polynomials* $\{x^i\}_{i\geq 0}$*. If* $\boldsymbol{\sigma}(\mathbf{x}) = h_n(\mathbf{x})$ *where* $\mathbf{x}$ *are on the surface of a unit sphere, then:*

$$\ddot{\boldsymbol{\sigma}}(\mathbf{x}_i{}^\intercal \mathbf{x}_j) = (\mathbf{x}_i{}^\intercal \mathbf{x}_j)^n \tag{E.27}$$

for $\mathbf{x}_i{}^\intercal \mathbf{x}_j \in [-1, 1]$.

The proof is given in [ODo21, p. 336], and also be found in [Rad+22b]. And so the kernel in eqn (E.25) becomes:

$$\kappa(\mathbf{x}_\mathsf{i}, \mathbf{x}_\mathsf{j}) = \frac{1}{\pi}\left(\langle \mathbf{x}_\mathsf{i}, \mathbf{x}_\mathsf{j}\rangle\left(\pi - cos^{-1}\langle \mathbf{x}_\mathsf{i}, \mathbf{x}_\mathsf{j}\rangle\right) + \sqrt{1 - \langle \mathbf{x}_\mathsf{i}, \mathbf{x}_\mathsf{j}\rangle^2}\right), \tag{E.28}$$

and we see the substitution of $\langle \mathbf{x}, \mathbf{x}\rangle = 1$ [Rad+22b].

Remark E.4.3. *(Double Descent Phenomenon and Over Parameterization)* The most noteworthy aspect of running NNGP regression with ever-wider width of $\hat{\mathbf{B}}$ is that it captures the double-descent phenomenon frequently associated with deep ANNs.

That is the test error of the model continue to decrease long after training error have flat lined *if* the mode is sufficiently wide. This flies in the face of both conventional wisdom and traditional optimization theory, which states that high capacity model tend to over fit data by simply interpolating the training points, and therefore cannot generalize well (fig (**??**)). In ANNs, interpolation leads to generalization (see fig E.10).

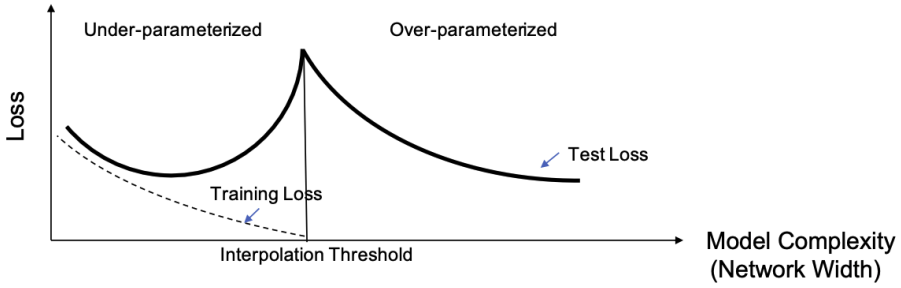


Figure E.10: Classical machine learning theory is allergic to the notion of interpolation, roughly defined as selecting a model whereby the number of parameters is equal or greater than the number of data points. For the model in eqn (E.18), interpolation threshold occurs at $\mathrm{d}_1 = m$ for m training examples. Beyond this threshold, the model simply interpolates the points, which leads to over fitting. In the ANN setting however, the number of parameters may be greater than the number of data points. Moreover, as the width of the ANN expand beyond that of the interpolation threshold, test error continues to decrease. The test error is maximal at the threshold, a phenomenon that can be remedied by regularization via early stopping. This suggests that not only do large ANNs interpolate data, *interpolation leads to generalization* [[Bel21, p. 18],[Rad+22b]].

Remark E.4.4. *(ANN Gaussian Process Regression)* While kernalized least squares regression is an example of a parametric method, we can also use eqn (E.28) in non-parametric methods

such as $\mathcal{GP}$ regression. Unlike parametric methods, $\mathcal{GP}$ regression not only predicts the mean $\mathsf{y} = \mathbf{E}[f(\mathbf{x}; \mathbf{w})]$ value, but also outputs an uncertainty around the prediction. This is accomplished by sampling from the space of all functions at the test point $\mathbf{x}_t$ after posterior updates on the training data. In particular, the mean of the sampled functions is the least squares solution, while the variance around the sample is determined by the kernel matrix built from training and test data. Notation wise, given test data $\mathbf{x}_t$, and training data $(\mathbf{X}, \mathbf{y})$, the expected value of y is written:

$$f(\mathbf{x}_t; \mathbf{w}) \mid \mathbf{X}, \mathbf{y}, \mathbf{x}_t \sim \mathcal{GP}\Big(\mathbf{E}[f(\mathbf{x}; \mathbf{w})],\ \kappa(\mathbf{x}_\mathsf{i}, \mathbf{x}_t)\Big) \qquad \mathbf{x}_\mathsf{i} \in \mathbf{X}.$$

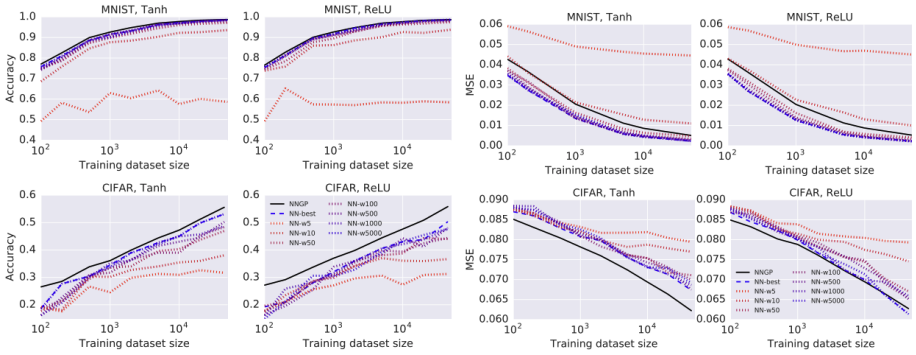
Here $\kappa(\cdot, \cdot)$ is the kernel function that defines the "uncertainty" of $\mathcal{GP}$ regression [RW06, p. 13]. [6] Now given training set $(\mathbf{X}, \mathbf{y})$ and test set $\mathbf{X}_t$, so that all $\mathbf{x}$'s are properly normalized and zero-meaned, we can construct kernel matrices from the data with:

$$\begin{aligned}
&\mathbf{K}_{\mathcal{GP}} = \begin{bmatrix} \mathbf{K} & \mathbf{K}_* \\ \mathbf{K}_*^\intercal & \mathbf{K}_{**} \end{bmatrix} + \sigma^2 \mathbf{I} \\
&\mathbf{K}_{\mathsf{ij}} = \kappa(\mathbf{x}_\mathsf{i}, \mathbf{x}_\mathsf{j}),\ \mathbf{x}_\mathsf{i}, \mathbf{x}_\mathsf{j} \in \mathbf{X} \\
&(\mathbf{K}_*)_{\mathsf{ij}} = \kappa(\mathbf{x}_\mathsf{i}, \mathbf{x}_\mathsf{j}),\ \mathbf{x}_\mathsf{i} \in \mathbf{X}, \mathbf{x}_\mathsf{j} \in \mathbf{X}_t, \qquad (\mathbf{K}_{**})_{\mathsf{ij}} = \kappa(\mathbf{x}_\mathsf{i}, \mathbf{x}_\mathsf{j}),\ \mathbf{x}_\mathsf{i} \in \mathbf{X}_t, \mathbf{x}_\mathsf{j} \in \mathbf{X}_t.
\end{aligned}$$

Here the ijth entry of every matrix is computed with the kernel κ defined in eqn (E.28). The constant σ is some normalization factor so that all the matrices are invertible. Next we can define the regression weights with $\mathbf{A} = (\mathbf{K} + \sigma^2 \mathbf{I})^{-1} \mathbf{y}$ per eqn (E.9). So the neural network $\mathcal{GP}$ regression is written:

$$\begin{aligned}
&f(\mathbf{X}_t) \mid \mathbf{X}, \mathbf{y}, \mathbf{X}_t \sim \mathcal{GP}(\boldsymbol{\mu}_p, \Sigma_p) \\
&\boldsymbol{\mu}_p = \mathbf{K}_*{}^\intercal (\mathbf{K} + \sigma^2 \mathbf{I})^{-1} \mathbf{y} \qquad \Sigma_p = (\mathbf{K}_{**} + \sigma^2 \mathbf{I}) - \mathbf{K}_*{}^\intercal (\mathbf{K} + \sigma^2 \mathbf{I})^{-1} \mathbf{K}_*.
\end{aligned} \tag{E.29}$$

Observe no training is required here, all that is needed is data fusion using the mean and kernel matrix expressions derived in Appendix **??**. In particular, the kernel function κ is defined entirely by eqn (E.28), and is constant during the equivalent ANN training path that would occur via gradient descent [LB18].



(a) Accuracy (b) Mean squared error

Figure E.11: Test accuracy and mean squared error on MNIST and CIFAR-10 dataset are shown for the best performing NNGP and best performing ANN for given width. 'NN-best' denotes the best performing (on the validation set) neural network across all widths and trials. $\mathcal{GP}$ regression often outperforms finite width networks. And as the width of ANNs increase, their performance approaches that of $\mathcal{GP}$ regression [LB18].

[6] $\mathcal{GP}$ regression is a large topic, a primer can be found in Appendix **??**. In particular, subsection "One Loop in Gaussian Process Regression" shows how to update regression mean and variance after data fusion.

Remark E.4.5. (*Alternate derivation of kernel function*) In eqn (E.20), the function κ is written using the definition of the kernel via eqn (E.10). There is an alternate derivation of the kernel from the expression of the ANN itself. Given the function $f(x;\mathbf{w}) = \mathbf{A}\boldsymbol{\sigma}(\hat{\mathbf{B}}x)$ with fixed $\hat{\mathbf{B}}$. And for simplicity sake, assume $x \in \mathbb{R}$, $\mathbf{X} \in \mathbb{R}^{1\times \mathsf{m}}$, $\mathbf{B} \in \mathbb{R}^{1\times 2}$, and $\mathbf{A} \in \mathbb{R}^{2\times 1}$. Then we can write $\mathbf{w}$ as:

$$\mathbf{w} = [\mathbf{A}_1, \mathbf{A}_2, \hat{\mathbf{B}}_1, \hat{\mathbf{B}}_2] = (\alpha_1, \alpha_2, \hat{b}_1, \hat{b}_2).$$

Now given some data point x_{i}, we can rewrite f with variable weights $\mathbf{w}$ at fixed x_{i} as:

$$f(\mathbf{w}; x_{\mathsf{i}}) = (\alpha_1, \alpha_2)\boldsymbol{\sigma}\bigg((\hat{b}_1, \hat{b}_2)^{\intercal} x_{\mathsf{i}}\bigg) = \alpha_1\boldsymbol{\sigma}(\hat{b}_1 x_{\mathsf{i}}) + \alpha_2\boldsymbol{\sigma}(\hat{b}_2 x_{\mathsf{i}}) = \sum_{\mathsf{r}=1}^{\mathsf{d}_1} \alpha_{\mathsf{r}}\boldsymbol{\sigma}(\hat{b}_{\mathsf{r}} x_{\mathsf{i}}).$$

Note this is stating f of $\mathbf{w}$ given *parameter* x, and the sum is over d_1, not the number of training points m. The concept should appear natural as it is just a variation of how Bayesian people express their ideas. The purpose of writing $f(\mathbf{w}; x_{\mathsf{i}})$ is to take the gradient of f w.r.t $\mathbf{w}$, not x. In gradient descent, we minimize this loss function:

$$\mathsf{loss}(\mathbf{w}) = \sum_{\mathsf{i}=1}^{\mathsf{m}} (\mathsf{y}_{\mathsf{i}} - f(\mathbf{w}; x_{\mathsf{i}}))^2.$$

Where $f(\hat{\mathbf{w}}; x_{\mathsf{i}})$ is initialized at some $\hat{\mathbf{w}}$, so that after some delta we have $\mathbf{w} = \hat{\mathbf{w}} + \delta$. This $f(\mathbf{w}; x_{\mathsf{i}})$ can be computed using the nth order Taylor expansion:

$$f(\mathbf{w}; x_{\mathsf{i}}) = f(\hat{\mathbf{w}}; x_{\mathsf{i}}) + \underbrace{\bigg\langle \nabla f(\hat{\mathbf{w}}; x_{\mathsf{i}}),\ (\mathbf{w} - \hat{\mathbf{w}}) \bigg\rangle}_{\text{gradient}} + \underbrace{\frac{1}{2}\bigg\langle (\mathbf{w} - \hat{\mathbf{w}}),\ \nabla^2 f(\hat{\mathbf{w}}; x_{\mathsf{i}})(\mathbf{w} - \hat{\mathbf{w}}) \bigg\rangle + \dots}_{\text{h.o.t}}$$

If we take the first order approximation with gradient only, and initialize training so that:

$$\hat{\mathbf{w}} = \mathbf{0} \qquad f(\hat{\mathbf{w}}; x_{\mathsf{i}}) = 0.$$

Then the first order approximation becomes:

$$f(\mathbf{w}; x_{\mathsf{i}}) = f(\hat{\mathbf{w}}; x_{\mathsf{i}}) + \bigg\langle \nabla f(\hat{\mathbf{w}}; x_{\mathsf{i}}),\ (\mathbf{w} - \hat{\mathbf{w}}) \bigg\rangle = \langle \nabla f(\hat{\mathbf{w}}; x_{\mathsf{i}}), \mathbf{w} \rangle.$$

The loss function over all the weights $\mathbf{w}$ is now rewritten:

$$\mathsf{loss}(\mathbf{w}) = \sum_{\mathsf{i}=1}^{\mathsf{m}} (\mathsf{y}_{\mathsf{i}} - f(\mathbf{w}; x_{\mathsf{i}}))^2 = \sum_{\mathsf{i}=1}^{\mathsf{m}} (\mathsf{y}_{\mathsf{i}} - \langle \underbrace{\nabla f(\hat{\mathbf{w}}; x_{\mathsf{i}})}_{\phi_{\mathbf{x}_{\mathsf{i}}}}, \mathbf{w} \rangle)^2 \overset{(1)}{=} \sum_{\mathsf{i}=1}^{\mathsf{m}} (\mathsf{y}_{\mathsf{i}} - \langle \phi_{\mathbf{x}_{\mathsf{i}}}, \mathbf{w} \rangle)^2.$$

Where going into (1) we note that $\nabla f(\hat{\mathbf{w}}; x_{\mathsf{i}})$ is in fact the corresponding feature map of x_{i}. So we can define the kernel using *just the gradient* with:

$$\kappa(x_{\mathsf{i}}, x_{\mathsf{j}}) = \phi_{\mathbf{x}_{\mathsf{i}}}{}^{\intercal} \phi_{\mathbf{x}_{\mathsf{j}}} = \underbrace{\bigg\langle \nabla f(\mathbf{w}; x_{\mathsf{i}}),\ \nabla f(\mathbf{w}; x_{\mathsf{j}}) \bigg\rangle}_{\text{neural tangent kernel}}.$$

Now we evaluate this kernel κ by considering $\nabla f(\mathbf{w}; x_{\mathsf{i}})$. Since $\hat{b}_1, \hat{b}_2$ are constant, we know $\frac{\partial f(\mathbf{w}; x_{\mathsf{i}})}{\partial \hat{b}_1} = 0$ and similarly for $\hat{b}_2$. So we simply take the gradient of f w.r.t $\mathbf{A}$ with:

$$\nabla_{\mathbf{A}} f(\mathbf{w}; x_{\mathsf{i}}) = \frac{\partial f(\mathbf{w}; x_{\mathsf{i}})}{\partial \mathbf{A}} = \bigg(\frac{\partial f(\mathbf{w}; x_{\mathsf{i}})}{\partial \alpha_1},\ \frac{\partial f(\mathbf{w}; x_{\mathsf{i}})}{\partial \alpha_2} \bigg) \tag{E.30}$$

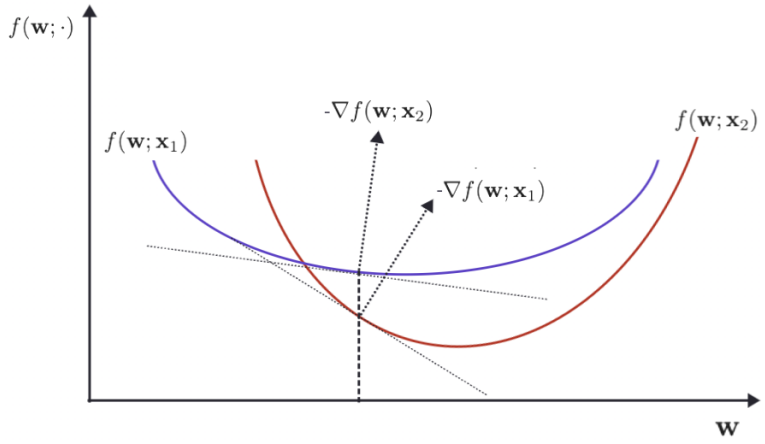


Figure E.12: Given two constants $\mathbf{x}_1$ and $\mathbf{x}_2$, two different functions of $f(\mathbf{w};\mathbf{x}_1)$ and $f(\mathbf{w};\mathbf{x}_2)$ can be drawn over the domain $\mathbf{w}$ of f. For every $\mathbf{w}$, the two functions have different gradients with $\nabla f(\mathbf{w};\mathbf{x}_1)$ and $\nabla f(\mathbf{w};\mathbf{x}_2)$. Their inner product is the neural tangent kernel, written: $\kappa(\mathbf{x}_1,\mathbf{x}_2) = \langle \nabla f(\mathbf{w};\mathbf{x}_1), \nabla f(\mathbf{w};\mathbf{x}_2)\rangle$.

$$= \Big(\boldsymbol{\sigma}(\hat{b}_1 x_{\mathrm{i}}),\ \boldsymbol{\sigma}(\hat{b}_2 x_{\mathrm{i}})\Big) = \boldsymbol{\sigma}(\hat{\mathbf{B}} x_{\mathrm{i}}), \tag{E.31}$$

and similarly for $\nabla f(\mathbf{w}; x_{\mathrm{j}})$. Now note that in this case, all higher order terms (h.o.t) disappear as we have $\nabla^2_{\mathbf{A}} f(\hat{\mathbf{w}}; x_{\mathrm{j}}) = 0$, e.t.c. Thus the first order Taylor expansion of f is sufficient to define the kernel with:

$$\kappa(x_{\mathrm{i}}, x_{\mathrm{j}}) = \Big\langle \nabla f(\mathbf{w}; x_{\mathrm{i}}),\ \nabla f(\mathbf{w}; x_{\mathrm{j}}) \Big\rangle \sim \Big\langle \boldsymbol{\sigma}(\hat{\mathbf{B}} x_{\mathrm{i}}),\ \boldsymbol{\sigma}(\hat{\mathbf{B}} x_{\mathrm{j}}) \Big\rangle = \langle \phi_{\mathbf{x}_{\mathrm{i}}}, \phi_{\mathbf{x}_{\mathrm{j}}} \rangle,$$

which is what eqn (E.20) asserts. The point here is that the kernel of f can be derived by taking the gradient of f w.r.t $\mathbf{w}$, defined at two data points $\mathbf{x}_{\mathrm{i}}$ and $\mathbf{x}_{\mathrm{j}}$. This intuition will lead into the discussion for neural tangent kernel, see fig (E.12) for this remark in picture form.

Neural Tangent Kernel for Feed-Forward ANNs

The neural tangent kernel (NTK) is a natural extension of the NNGP of the previous section. However now the we will consider two-layer feed-forward ANNs where all the parameters are learned. Recall when $\hat{\mathbf{B}}$ is fixed, the model $f(\mathbf{x};\mathbf{w}) = \mathbf{A}\boldsymbol{\sigma}(\hat{\mathbf{B}}\mathbf{x})$ reduces to a kernel regression problem so that:

1. the objective is convex;
2. by definition of eqn (E.10), we know the kernel is defined with $\kappa(\mathbf{x}_{\mathrm{i}}, \mathbf{x}_{\mathrm{j}}) = \langle \phi_{\mathbf{x}_{\mathrm{i}}}, \phi_{\mathbf{x}_{\mathrm{j}}} \rangle$, where $\phi_{\mathbf{x}_{\mathrm{i}}} = \boldsymbol{\sigma}(\hat{\mathbf{B}}\mathbf{x}_{\mathrm{i}})$.

In the case where both $(\mathbf{A}, \mathbf{B})$ are variable, 1) the objective is no longer convex, and 2) defining the feature map ϕ is more complicated, as during training the values of ϕ changes w.r.t regression parameters $\mathbf{A}$. The following proposition shows that in the limiting case where the width of $\mathbf{B}$ goes to infinity, there is a constant neural tangent kernel that can be derived even if all the ANN parameters are variable.

Proposition E.4.6. *The setting is similar to the previous section. Given:*

$$
\begin{aligned}
&f : \mathbb{R}^{d_0} \to \mathbb{R} \\
&f(\mathbf{x}; \mathbf{w}) = \frac{1}{\sqrt{d_1}} \mathbf{A}\boldsymbol{\sigma}(\mathbf{B}\mathbf{x}) \qquad \text{for } \mathbf{y} \in \mathbb{R} \;\; \mathbf{x} \in \mathbb{R}^{d_0} \text{ with } \mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \\
&\text{where } \mathbf{A} \in \mathbb{R}^{\mathbf{1} \times d_1} \qquad \mathbf{B} \in \mathbb{R}^{d_1 \times d_0} \qquad \mathbf{A}_{ij}, \mathbf{B}_{ij} \overset{\text{iid}}{\sim} \mathcal{N}(0, 1) \\
&\boldsymbol{\sigma}(\cdot) : \mathbb{R} \to \mathbb{R} \qquad \ddot{\boldsymbol{\sigma}} : [-1, 1] \to \mathbb{R}.
\end{aligned}
$$

Then as $d_1 \to \infty$, the neural tangent kernel (NTK) is defined with:

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \ddot{\boldsymbol{\sigma}}(\mathbf{x}_i{}^\intercal \mathbf{x}_j) + \ddot{\boldsymbol{\sigma}}'(\mathbf{x}_i{}^\intercal \mathbf{x}_j)(\mathbf{x}_i{}^\intercal \mathbf{x}_j), \tag{E.32}$$

where $\ddot{\boldsymbol{\sigma}}'(\cdot)$ is the first derivative of $\ddot{\boldsymbol{\sigma}}$ w.r.t the terms inside.

Proof. This proof contains two parts: 1) using the gradient approximation of f to derive expression E.32; and 2) argue that the gradient is sufficient to express κ, since the Hessian goes to zero as the width of $\mathbf{B}$ increases. See fig (**??**) for intuition for the NTK.

1. Similar to remark E.4.5, we place all the parameters of $(\mathbf{A}, \mathbf{B})$ into a weights matrix of dimension $\mathrm{d}_1 + \mathrm{d}_1 \times \mathrm{d}_0$:

$$\mathbf{w} = [\mathbf{A}_{11}, \ldots, \mathbf{A}_{1\mathrm{d}_1}, \mathbf{B}_{11}, \ldots, \mathbf{B}_{\mathrm{d}_0\mathrm{d}_1}] = [\alpha_1, \ldots, \alpha_{\mathrm{d}_1}, b_{11}, \ldots, b_{\mathrm{d}_0\mathrm{d}_1}], \tag{E.33}$$

   so that we have $f(\mathbf{w}; \mathbf{x}) : \mathbb{R}^{\mathrm{d}_1 + \mathrm{d}_0\mathrm{d}_1} \to \mathbb{R}$. Similar to eqn (E.21), we can write $f(\mathbf{w}; \mathbf{x})$ with dual activation $\ddot{\boldsymbol{\sigma}}$ as:

$$f(\mathbf{w}; \mathbf{x}) = \frac{1}{\sqrt{\mathrm{d}_1}} \sum_{\mathsf{r}=1}^{\mathrm{d}_1} \alpha_{\mathsf{r}} \ddot{\boldsymbol{\sigma}}(\langle \mathbf{b}_{\mathsf{r}}, \mathbf{x} \rangle) = \frac{1}{\sqrt{\mathrm{d}_1}} \Big(\alpha_1 \ddot{\boldsymbol{\sigma}}(\mathbf{b}_1{}^\intercal \mathbf{x}) + \ldots + \alpha_{\mathrm{d}_1} \ddot{\boldsymbol{\sigma}}(\mathbf{b}_{\mathrm{d}_1}{}^\intercal \mathbf{x}) \Big).$$

   Dropping the $1/\sqrt{\mathrm{d}_1}$ for ease of notation, we take the gradient w.r.t $\{\alpha_{\mathsf{r}}\}_{r \le \mathrm{d}_1}$:

$$\nabla_{\alpha_{\mathsf{r}}} f(\mathbf{w}; \mathbf{x}) \sim \frac{\partial}{\partial \alpha_{\mathsf{r}}} \Big(\ldots + \alpha_{\mathsf{r}} \ddot{\boldsymbol{\sigma}}(\mathbf{b}_{\mathsf{r}}{}^\intercal \mathbf{x}) + \ldots \Big) = \ddot{\boldsymbol{\sigma}}(\mathbf{b}_{\mathsf{r}}{}^\intercal \mathbf{x}). \tag{E.34}$$

   Taking the gradient w.r.t $\{b_{\mathsf{rc}}\}_{r \le \mathrm{d}_1, c \le \mathrm{d}_0}$:

$$\nabla_{b_{\mathsf{rc}}} f(\mathbf{w}; \mathbf{x}) \sim \frac{\partial}{\partial b_{\mathsf{rc}}} \Big(\ldots + \alpha_{\mathsf{r}} \ddot{\boldsymbol{\sigma}}(\mathbf{b}_{\mathsf{r}}{}^\intercal \mathbf{x}) + \ldots \Big) = \alpha_{\mathsf{r}} \ddot{\boldsymbol{\sigma}}'(\mathbf{b}_{\mathsf{r}}{}^\intercal \mathbf{x}) x_c.$$

   Putting all the terms together, the gradient of f becomes:

$$\nabla f(\mathbf{w}; \mathbf{x}) = \frac{1}{\sqrt{\mathrm{d}_1}} \Big(\ddot{\boldsymbol{\sigma}}(\mathbf{b}_1{}^\intercal \mathbf{x}), \ldots, \ddot{\boldsymbol{\sigma}}(\mathbf{b}_{\mathrm{d}_1}{}^\intercal \mathbf{x}), \;\; \alpha_1 \ddot{\boldsymbol{\sigma}}'(\mathbf{b}_1{}^\intercal \mathbf{x}) x_1, \;\; \ldots \;\; , \alpha_{\mathrm{d}_1} \ddot{\boldsymbol{\sigma}}'(\mathbf{b}_{\mathrm{d}_1}{}^\intercal \mathbf{x}) x_{\mathrm{d}_0} \Big).$$

   Then the kernel for some fixed value of d_1 becomes:

$$
\begin{aligned}
\kappa(\mathbf{x}_{\mathsf{i}}, \mathbf{x}_{\mathsf{j}}) &= \Big\langle \nabla f(\mathbf{w}; \mathbf{x}_{\mathsf{i}}), \; \nabla f(\mathbf{w}; \mathbf{x}_{\mathsf{j}}) \Big\rangle \\
&= \underbrace{\frac{1}{\mathrm{d}_1} \sum_{\mathsf{r}=1}^{\mathrm{d}_1} \ddot{\boldsymbol{\sigma}}(\mathbf{b}_{\mathsf{r}}{}^\intercal \mathbf{x}_{\mathsf{i}}) \cdot \ddot{\boldsymbol{\sigma}}(\mathbf{b}_{\mathsf{r}}{}^\intercal \mathbf{x}_{\mathsf{j}})}_{\text{NNGP}} + \underbrace{\frac{1}{\mathrm{d}_1} \sum_{\mathsf{r}=1}^{\mathrm{d}_1} \sum_{\mathsf{c}=1}^{\mathrm{d}_0} \alpha_{\mathsf{r}}^2 \cdot \ddot{\boldsymbol{\sigma}}'(\mathbf{b}_{\mathsf{r}}{}^\intercal \mathbf{x}_{\mathsf{i}}) \cdot \ddot{\boldsymbol{\sigma}}'(\mathbf{b}_{\mathsf{r}}{}^\intercal \mathbf{x}_{\mathsf{j}}) \cdot x_{\mathsf{ic}} x_{\mathsf{jc}}}_{\text{contribution from changing } \mathbf{B}}.
\end{aligned}
$$

   Observe the first term under-bracketed is just the NNGP, which was derived in remark E.4.5. The second term with the double sums corresponds to contributions from the feature matrix $\mathbf{B}$ that is also updated via gradient descent.

Now we take the width of $\mathbf{B}$ to infinity with $\mathrm{d}_1 \to \infty$, the NNGP part becomes:

$$\lim_{\mathrm{d}_1 \to \infty} \left\langle \ddot{\boldsymbol{\sigma}}(\mathbf{B}\mathbf{x}_\mathsf{i}),\ \ddot{\boldsymbol{\sigma}}(\mathbf{B}\mathbf{x}_\mathsf{j}) \right\rangle = \ddot{\boldsymbol{\sigma}}(\mathbf{x}_\mathsf{i}^\intercal \mathbf{x}_\mathsf{j}).$$

The equality follows as we are working with the dual activation with expectation and variance defined in eqn (E.26). Next we consider the contribution from $\mathbf{B}$:

$$\begin{aligned}
&\lim_{\mathrm{d}_1 \to \infty} \frac{1}{\mathrm{d}_1} \sum_{\mathsf{r}=1}^{\mathrm{d}_1} \sum_{\mathsf{c}=1}^{\mathrm{d}_0} \alpha_\mathsf{r}^2 \cdot \ddot{\boldsymbol{\sigma}}'(\mathbf{b}_\mathsf{r}^\intercal \mathbf{x}_\mathsf{i}) \cdot \ddot{\boldsymbol{\sigma}}'(\mathbf{b}_\mathsf{r}^\intercal \mathbf{x}_\mathsf{j}) \cdot x_{\mathsf{i}\mathsf{c}} x_{\mathsf{j}\mathsf{c}} \\
&\quad = \lim_{\mathrm{d}_1 \to \infty} \frac{1}{\mathrm{d}_1} \sum_{\mathsf{r}=1}^{\mathrm{d}_1} \alpha_\mathsf{r}^2 \cdot \ddot{\boldsymbol{\sigma}}'(\mathbf{b}_\mathsf{r}^\intercal \mathbf{x}_\mathsf{i}) \cdot \ddot{\boldsymbol{\sigma}}'(\mathbf{b}_\mathsf{r}^\intercal \mathbf{x}_\mathsf{j}) \sum_{\mathsf{c}=1}^{\mathrm{d}_0} x_{\mathsf{i}\mathsf{c}} x_{\mathsf{j}\mathsf{c}} \\
&\quad = \lim_{\mathrm{d}_1 \to \infty} \frac{1}{\mathrm{d}_1} \sum_{\mathsf{r}=1}^{\mathrm{d}_1} \alpha_\mathsf{r}^2 \cdot \ddot{\boldsymbol{\sigma}}'(\mathbf{b}_\mathsf{r}^\intercal \mathbf{x}_\mathsf{i}) \cdot \ddot{\boldsymbol{\sigma}}'(\mathbf{b}_\mathsf{r}^\intercal \mathbf{x}_\mathsf{j}) \langle \mathbf{x}_\mathsf{i}, \mathbf{x}_\mathsf{j} \rangle \\
&\quad = \mathbf{x}_\mathsf{i}^\intercal \mathbf{x}_\mathsf{j} \underbrace{\lim_{\mathrm{d}_1 \to \infty} \frac{1}{\mathrm{d}_1} \sum_{\mathsf{r}=1}^{\mathrm{d}_1} \alpha_\mathsf{r}^2 \ddot{\boldsymbol{\sigma}}'(\mathbf{b}_\mathsf{r}^\intercal \mathbf{x}_\mathsf{i}) \cdot \ddot{\boldsymbol{\sigma}}'(\mathbf{b}_\mathsf{r}^\intercal \mathbf{x}_\mathsf{j})}_{\text{NNGP w/ } \ddot{\boldsymbol{\sigma}}'(\cdot)}.
\end{aligned}$$

Now we examine NNGP w/ $\ddot{\boldsymbol{\sigma}}'(\cdot)$ as $\mathrm{d}_1 \to \infty$, consider:

$$\begin{aligned}
&\lim_{\mathrm{d}_1 \to \infty} \frac{1}{\mathrm{d}_1} \sum_{\mathsf{r}=1}^{\mathrm{d}_1} \alpha_\mathsf{r}^2 \ddot{\boldsymbol{\sigma}}'(\mathbf{b}_\mathsf{r}^\intercal \mathbf{x}_\mathsf{i}) \cdot \ddot{\boldsymbol{\sigma}}'(\mathbf{b}_\mathsf{r}^\intercal \mathbf{x}_\mathsf{j}) \xrightarrow{\mathfrak{p}} \\
&\quad \overset{(1)}{=} \mathbf{E}_{\alpha_\mathsf{r} \sim \mathcal{N}(0,1), \mathbf{B} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \left[\alpha_\mathsf{r}^2 \left\langle \ddot{\boldsymbol{\sigma}}'(\mathbf{B}\mathbf{x}_\mathsf{i}),\ \ddot{\boldsymbol{\sigma}}'(\mathbf{B}\mathbf{x}_\mathsf{j}) \right\rangle \right] \\
&\quad \overset{(2)}{=} \mathbf{E}_{\alpha_\mathsf{r} \sim \mathcal{N}(0,1)} \left[\alpha_\mathsf{r}^2 \right] \cdot \underbrace{\mathbf{E}_{\mathbf{B} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \left[\left\langle \ddot{\boldsymbol{\sigma}}'(\mathbf{B}\mathbf{x}_\mathsf{i}),\ \ddot{\boldsymbol{\sigma}}'(\mathbf{B}\mathbf{x}_\mathsf{j}) \right\rangle \right]}_{\text{r.h.s}} \\
&\quad \overset{(3)}{=} \ddot{\boldsymbol{\sigma}}'(\mathbf{x}_\mathsf{i}^\intercal \mathbf{x}_\mathsf{j}).
\end{aligned}$$

Where going into (1) is the consequence of the limit. Going into (2) we use the fact that α_r and $\mathbf{B}$ are independent variables. Going into (3) we note that $\mathbf{E}_{\alpha_\mathsf{r} \sim \mathcal{N}(0,1)}[\alpha_\mathsf{r}^2] = 1$, while the r.h.s is $\ddot{\boldsymbol{\sigma}}'(\mathbf{x}_\mathsf{i}^\intercal \mathbf{x}_\mathsf{j})$ by the same argument as E.26. Putting the above two expression together and we have the first order approximation of the kernel as: $\kappa(\mathbf{x}_\mathsf{i}, \mathbf{x}_\mathsf{j}) = \ddot{\boldsymbol{\sigma}}(\mathbf{x}_\mathsf{i}^\intercal \mathbf{x}_\mathsf{j}) + \ddot{\boldsymbol{\sigma}}'(\mathbf{x}_\mathsf{i}^\intercal \mathbf{x}_\mathsf{j})(\mathbf{x}_\mathsf{i}^\intercal \mathbf{x}_\mathsf{j})$ in the limit as expected.

Example E.4.7. *(NTK for ReLU)* We can complete the computation of NTK for a specific $\ddot{\boldsymbol{\sigma}}$, namely the ReLU activation. Define appropriately normalized f with $f(\mathbf{x}; \mathbf{w}) = \frac{\sqrt{2}}{\sqrt{\mathrm{d}_1}} \mathbf{A}\boldsymbol{\sigma}(\mathbf{B}\mathbf{x})$. [7] Its derivative is given by:

$$\ddot{\boldsymbol{\sigma}}'(\mathbf{x}_\mathsf{i}^\intercal \mathbf{x}_\mathsf{j}) = \frac{1}{\pi}(\pi - cos^{-1}(\mathbf{x}_\mathsf{i}^\intercal \mathbf{x}_\mathsf{j})).$$

And so we have:

$$\begin{aligned}
\kappa(\mathbf{x}_\mathsf{i}, \mathbf{x}_\mathsf{j}) &= \ddot{\boldsymbol{\sigma}}(\mathbf{x}_\mathsf{i}^\intercal \mathbf{x}_\mathsf{j}) + \ddot{\boldsymbol{\sigma}}'(\mathbf{x}_\mathsf{i}^\intercal \mathbf{x}_\mathsf{j})(\mathbf{x}_\mathsf{i}^\intercal \mathbf{x}_\mathsf{j}) \\
&= \frac{1}{\pi} \left(\mathbf{x}_\mathsf{i}^\intercal \mathbf{x}_\mathsf{j} \left(\pi - cos^{-1}(\mathbf{x}_\mathsf{i}^\intercal \mathbf{x}_\mathsf{j}) \right) + \sqrt{1 - (\mathbf{x}_\mathsf{i}^\intercal \mathbf{x}_\mathsf{j})^2} \right) \\
&\quad + \frac{\mathbf{x}_\mathsf{i}^\intercal \mathbf{x}_\mathsf{j}}{\pi} (\pi - cos^{-1}(\mathbf{x}_\mathsf{i}^\intercal \mathbf{x}_\mathsf{j})),
\end{aligned} \tag{E.35}$$

by using the prior expression for $\ddot{\boldsymbol{\sigma}}(\mathbf{x}_\mathsf{i}^\intercal \mathbf{x}_\mathsf{j})$ from E.28 [Rad+22d].

[7] The $\sqrt{2}$ is the normalization constant for $\ddot{\boldsymbol{\sigma}}$.

2. In part two, we further the argument by stating that the first order approximation to the kernel in eqn (E.35) is sufficient. Recall in remark E.4.5, the Hessian is zero since $\nabla^2_{\mathbf{A}}\boldsymbol{\sigma}(\hat{\mathbf{B}}\mathbf{x}) = 0$ as $\hat{\mathbf{B}}$ is a constant by definition. In this case the $\mathbf{B}$ is variable, yet the claim is that:

   a) The Hessian $\nabla^2(\cdot)$ w.r.t the weights approaches zero as $\mathrm{d}_1 \to \infty$, written:

   $$\|\nabla^2(f(\mathbf{w};\mathbf{x}))\|_2 = \mathcal{O}\Big(\frac{1}{\sqrt{\mathrm{d}_1}}\Big). \tag{E.36}$$

   b) So that given any particular training path via gradient descent, the neural tangent kernel is constant:

   $$|\kappa(\mathbf{x}_\mathrm{i}, \mathbf{x}_\mathrm{j})| = \mathcal{O}(1). \tag{E.37}$$

   c) Thus the model f "transitions to linearity" as $\mathrm{d}_1 \to \infty$ [LZB21], expressed by:

   $$f(\mathbf{w};\mathbf{x}) = f(\hat{\mathbf{w}};\mathbf{x}) + \Big\langle \nabla_{\mathbf{w}} f(\hat{\mathbf{w}};\mathbf{x}),\ (\mathbf{w}-\hat{\mathbf{w}}) \Big\rangle. \tag{E.38}$$

   Similar the gradient case, we have $f : \mathbb{R}^{\mathrm{d}_1+\mathrm{d}_0\mathrm{d}_1} \to \mathbb{R}$, whose input ranges over the weights defined in E.33: $\mathbf{w} = [\alpha_1, \dots, \alpha_{\mathrm{d}_1}, b_{11}, \dots, b_{\mathrm{d}_0\mathrm{d}_1}]$. The Hessian then is a square matrix of dimension $(\mathrm{d}_1 + \mathrm{d}_0\mathrm{d}_1)^2$, and can be expressed in 2×2 block format:

$$\nabla^2(f(\mathbf{w};\mathbf{x})) = \left[\begin{array}{ccc|ccc} \frac{\partial^2 f}{\partial\alpha_1\partial\alpha_1} & \cdots & \frac{\partial^2 f}{\partial\alpha_1\partial\alpha_{\mathrm{d}_1}} & \frac{\partial^2 f}{\partial\alpha_1\partial b_{11}} & \cdots & \frac{\partial^2 f}{\partial\alpha_1\partial b_{\mathrm{d}_0\mathrm{d}_1}} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial\alpha_{\mathrm{d}_1}\partial\alpha_1} & \cdots & \frac{\partial^2 f}{\partial\alpha_{\mathrm{d}_1}\partial\alpha_{\mathrm{d}_1}} & \frac{\partial^2 f}{\partial\alpha_{\mathrm{d}_1}\partial b_{11}} & \cdots & \frac{\partial^2 f}{\partial\alpha_{\mathrm{d}_1}\partial b_{\mathrm{d}_0\mathrm{d}_1}} \\ \hline \frac{\partial^2 f}{\partial b_{11}\partial\alpha_1} & \cdots & \frac{\partial^2 f}{\partial b_{11}\partial\alpha_{\mathrm{d}_1}} & \frac{\partial^2 f}{\partial b_{11}\partial b_{11}} & \cdots & \frac{\partial^2 f}{\partial b_{11}\partial b_{\mathrm{d}_0\mathrm{d}_1}} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial b_{\mathrm{d}_0\mathrm{d}_1}\partial\alpha_1} & \cdots & \frac{\partial^2 f}{\partial b_{\mathrm{d}_0\mathrm{d}_1}\partial\alpha_{\mathrm{d}_1}} & \frac{\partial^2 f}{\partial b_{\mathrm{d}_0\mathrm{d}_1}\partial b_{11}} & \cdots & \frac{\partial^2 f}{\partial b_{\mathrm{d}_0\mathrm{d}_1}\partial b_{\mathrm{d}_0\mathrm{d}_1}} \end{array}\right].$$

   Recall the function is of form $f(\mathbf{w};\mathbf{x}) \sim \sum_{\mathsf{r}=1}^{\mathrm{d}_1} \alpha_\mathsf{r}\ddot{\boldsymbol{\sigma}}(\langle \mathbf{b}_\mathsf{r}, \mathbf{x}\rangle)$. Now observe:

   a) *Top left block*: all the terms are zero as it is taking the double derivative w.r.t. α_r:

   $$\frac{\partial^2}{\partial\alpha_\mathsf{r}\partial\alpha_s}(\dots + \alpha_\mathsf{r}\ddot{\boldsymbol{\sigma}}(\mathbf{b}_\mathsf{r}{}^\intercal\mathbf{x}) + \dots) = 0.$$

   b) *Top right block*: all non-diagonal terms are zero. Along the diagonal we have:

   $$\frac{\partial^2}{\partial\alpha_\mathsf{r}\partial b_{rr}}(\dots + \alpha_\mathsf{r}\ddot{\boldsymbol{\sigma}}(\mathbf{b}_\mathsf{r}{}^\intercal\mathbf{x}) + \dots) = \frac{\partial}{\partial b_{rr}}\ddot{\boldsymbol{\sigma}}(\mathbf{b}_\mathsf{r}{}^\intercal\mathbf{x}) = \ddot{\boldsymbol{\sigma}}'(\mathbf{b}_\mathsf{r}{}^\intercal\mathbf{x})x_r.$$

   c) *Bottom left block*: this is the same as the top right block. So the non-diagonal terms are zero, and all the diagonal terms are the same as above.

   d) *Bottom right block*: the non-diagonal terms are zero again. Along the diagonal we have:

   $$\frac{\partial^2}{\partial b_{rr}\partial b_{rr}}(\dots + \alpha_\mathsf{r}\ddot{\boldsymbol{\sigma}}(\mathbf{b}_\mathsf{r}{}^\intercal\mathbf{x}) + \dots) = \frac{\partial}{\partial b_{rr}}\alpha_\mathsf{r}\ddot{\boldsymbol{\sigma}}'(\mathbf{b}_\mathsf{r}{}^\intercal\mathbf{x})x_\mathsf{r} = \alpha_\mathsf{r}\ddot{\boldsymbol{\sigma}}''(\mathbf{b}_\mathsf{r}{}^\intercal\mathbf{x})x_\mathsf{r}^2.$$

All in all, the Hessian is sparse: where the top left block are all zeros, and in the other three blocks only the diagonal terms have non-zero values. Now the spectral norm of the Hessian can be written:

$$\|\nabla^2(f(\mathbf{w};\mathbf{x}))\|_2 = \frac{1}{\sqrt{\mathrm{d}_1}}\max_{r\in \mathrm{d}_1}\left|\frac{\alpha_r\ddot{\boldsymbol{\sigma}}''(\mathbf{b}_r{}^\intercal\mathbf{x})x_r^2 + \sqrt{\alpha_r^2\ddot{\boldsymbol{\sigma}}''(\mathbf{b}_r{}^\intercal\mathbf{x})x_r^4 + 4\ddot{\boldsymbol{\sigma}}'(\mathbf{b}_r{}^\intercal\mathbf{x})^2x_r^2}}{2}\right|$$
$$\overset{(1)}{=} \frac{C}{\sqrt{\mathrm{d}_1}} = \mathcal{O}\Big(\frac{1}{\sqrt{\mathrm{d}_1}}\Big),$$

where going into (1), we use the fact that both $\mathbf{b}_r$ and $\mathbf{x}$ are bounded (see [LZB21, p. 6], and [Rad+22d, p. 8] for eigenvalue computation). Next consider claim E.37, the kernel can be written with ANN parameters:

$$\kappa(\mathbf{x}_\mathrm{i},\mathbf{x}_\mathrm{j}) = \frac{1}{\mathrm{d}_1}\Big(\sum_{r=1}^{\mathrm{d}_1}\ddot{\boldsymbol{\sigma}}(\mathbf{b}_r{}^\intercal\mathbf{x}_\mathrm{i})\cdot\ddot{\boldsymbol{\sigma}}(\mathbf{b}_r{}^\intercal\mathbf{x}_\mathrm{j}) \;+\; \mathbf{x}_\mathrm{i}{}^\intercal\mathbf{x}_\mathrm{j}\sum_{r=1}^{\mathrm{d}_1}\alpha_r^2\cdot\ddot{\boldsymbol{\sigma}}'(\mathbf{b}_r{}^\intercal\mathbf{x}_\mathrm{i})\cdot\ddot{\boldsymbol{\sigma}}'(\mathbf{b}_r{}^\intercal\mathbf{x}_\mathrm{j})\Big).$$

Observe the kernel value is normalized by $1/\mathrm{d}_1$, thus it remains constant as $\mathrm{d}_1\to\infty$. Therefore $|\kappa(\mathbf{x}_\mathrm{i},\mathbf{x}_\mathrm{j})| = \mathcal{O}(1)$, while Hessian of f approaches 0 as $\mathrm{d}_1\to\infty$. All in all, this implies the NTK becomes constant, and the the gradient is sufficient to characterize $f(\mathbf{w};\mathbf{x})$ as E.38 states. Thus solving ANN with stochastic gradient descent is equivalent to solving kernel regression with the NTK [[LZB21] [Rad+22d]].

■

Remark E.4.8. (*Limitations of proposition E.4.6*) Note the function in E.4.6 is of form $f(\mathbf{x};\mathbf{w}) = \mathbf{A}\boldsymbol{\sigma}(\mathbf{B}\mathbf{x})$, that is it does not have nonlinear activation function applied to the output.If however f is defined such that $f(\mathbf{x};\mathbf{w}) = \boldsymbol{\sigma}(\mathbf{A}\boldsymbol{\sigma}(\mathbf{B}\mathbf{x}))$, then expression E.38 would fail, as the Hessian elements would have additional terms from the nonlinear transformation on the output of the regression. That is to say the Hessian does not tend to zero as $\mathrm{d}_1\to\infty$, and the NTK does not stay constant throughout the training path. This observation rules out some ANNs from the analysis framework developed thus far, i.e. ones where the last output layer is logistic regression. Similarly, if a narrow "bottleneck" layer is added to the ANN, then even if the bottle neck layer is linear, the Hessian term would be large compared to the gradient term, and transition to linearity of the ANN would be lost. This is important as in many ANN architectures, "down-sampling" layers are interspersed throughout to "shake out" any spurious information that may accumulate by projecting the information onto higher dimensions [LZB21].

NTK for Deep Feed-Forward ANNs

This section derives the NNGP and NTK for fully connected, feed-forward ANNs.

Proposition E.4.9. *(NTK for Fully Connected Networks) Let f be an ANN with L hidden layers defined on $\mathbf{x}$'s and some $d_0 \geq 1$ s.t.:*

$$\mathbf{x}_i, \mathbf{x}_j \in \mathbb{R}^{d_0\times 1} \qquad f(\mathbf{x};\mathbf{w}) : \mathbb{R}^{d_0}\to\mathbb{R},$$

Now defined the $L+1$ weights and element-wise activation function:

$$\mathbf{w} = [(\mathbf{W}^{L+1})_r, .., (\mathbf{W}^l)_{rc}, .., (\mathbf{W}^1)_{rc}] \qquad \mathbf{W}^l\in\mathbb{R}^{d_l\times d_{l-1}} \qquad \mathbf{W}^{L+1}\in\mathbb{R}^{1\times d_L} \qquad \boldsymbol{\sigma}(\cdot):\mathbb{R}\to\mathbb{R},$$

where $0\leq l\leq L+1$. And similar to E.4.6, redefine the function so that it is parameterized by some specific $\mathbf{x}$:

$$f:\mathbb{R}^{|\mathbf{w}|}\to\mathbb{R}$$
$$f(\mathbf{w};\mathbf{x}) = \mathbf{W}^{L+1}\frac{1}{\sqrt{d_L}}\boldsymbol{\sigma}\Big(\mathbf{W}^L\frac{1}{\sqrt{d_{L-1}}}\boldsymbol{\sigma}\Big(\dots\mathbf{W}^2\frac{1}{\sqrt{d_1}}\boldsymbol{\sigma}(\mathbf{W}^1\mathbf{x})\dots\Big)\Big).$$

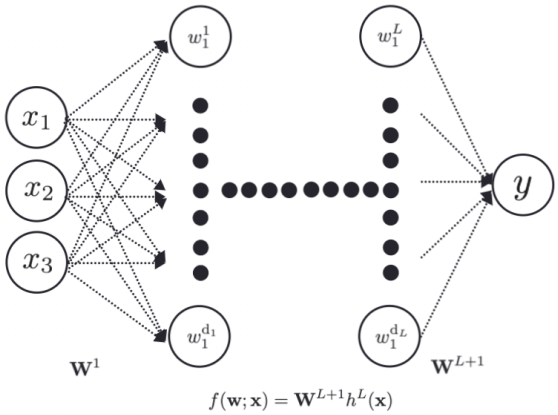


Figure E.13: Deep feed-forward ANN defined by function $f : \mathbb{R}^3 \to \mathbb{R}$, with $L+1$ layers $\mathbf{w} = [\mathbf{W}^1, \ldots, \mathbf{W}^L]$, and L hidden layers with functions $h^1(\mathbf{x}), \ldots, h^L(\mathbf{x})$.

Alternatively we can write the function above in recurrence form with hidden layer functions $\{h^l\}_{0\leq l\leq L}$*, so that the* 0*th layer is the data* $\mathbf{x} \in \mathbb{R}^{d_0\times 1}$*:*

$$f(\mathbf{w};\mathbf{x}) = \mathbf{W}^{L+1}h^L(\mathbf{w};\mathbf{x}) \qquad h^l(\mathbf{w};\mathbf{x}) = \frac{1}{\sqrt{d_l}}\boldsymbol{\sigma}(\mathbf{W}^l h^{l-1}(\mathbf{w};\mathbf{x})) \qquad h^0(\mathbf{w};\mathbf{x}) = \mathbf{x}. \quad \text{(E.39)}$$

Then as $d_1, \ldots, d_L \to \infty$ *in order, the NNGP and NTK are given by kernels* τ *and* κ*:*

$$\begin{aligned} &\tau^L(\mathbf{x}_i,\mathbf{x}_j) = \ddot{\boldsymbol{\sigma}}(\tau^{L-1}(\mathbf{x}_i,\mathbf{x}_j)) && \kappa^L(\mathbf{x}_i,\mathbf{x}_j) = \tau^L(\mathbf{x}_i,\mathbf{x}_j) + \kappa^{L-1}(\mathbf{x}_i,\mathbf{x}_j)\ddot{\boldsymbol{\sigma}}'(\kappa^{L-1}(\mathbf{x}_i,\mathbf{x}_j)) \\ &\tau^0(\mathbf{x}_i,\mathbf{x}_j) = \mathbf{x}_i{}^\intercal\mathbf{x}_j && \kappa^0(\mathbf{x}_i,\mathbf{x}_j) = \mathbf{x}_i{}^\intercal\mathbf{x}_j. \end{aligned} \quad \text{(E.40)}$$

Where the dual activation $\ddot{\boldsymbol{\sigma}}'(\cdot)$ *is defined so that* $\ddot{\boldsymbol{\sigma}}'(1) = 1$*.*

Proof. The proof is provided in [Rad+22c] and proceeds by induction.

1. *Base Case*: Let $L = 1$, this is the case in E.32. So the kernel is defined for the two layer feed-forward ANN, and we can directly write the kernels at the 0th (read: the data) and 1st layer with:

   $$\tau^0(\mathbf{x}_\mathrm{i},\mathbf{x}_\mathrm{j}) = \mathbf{x}_\mathrm{i}{}^\intercal\mathbf{x}_\mathrm{j} \qquad \tau^1(\mathbf{x}_\mathrm{i},\mathbf{x}_\mathrm{j}) = \ddot{\boldsymbol{\sigma}}(\mathbf{x}_\mathrm{i}{}^\intercal\mathbf{x}_\mathrm{j}) \qquad \kappa^0(\mathbf{x}_\mathrm{i},\mathbf{x}_\mathrm{j}) = \mathbf{x}_\mathrm{i}{}^\intercal\mathbf{x}_\mathrm{j}.$$

   And so similar to E.32 we have:

   $$\begin{aligned} \kappa^1(\mathbf{x}_\mathrm{i},\mathbf{x}_\mathrm{j}) &= \tau^1(\mathbf{x}_\mathrm{i},\mathbf{x}_\mathrm{j}) + \kappa^0(\mathbf{x}_\mathrm{i},\mathbf{x}_\mathrm{j})\ddot{\boldsymbol{\sigma}}'(\tau^0(\mathbf{x}_\mathrm{i},\mathbf{x}_\mathrm{j})) \\ &= \ddot{\boldsymbol{\sigma}}(\mathbf{x}_\mathrm{i}{}^\intercal\mathbf{x}_\mathrm{j}) + (\mathbf{x}_\mathrm{i}{}^\intercal\mathbf{x}_\mathrm{j})\ddot{\boldsymbol{\sigma}}'(\mathbf{x}_\mathrm{i}{}^\intercal\mathbf{x}_\mathrm{j}), \end{aligned} \quad \text{(E.41)}$$

   per claim $E.40$.

2. *Inductive Case*: let $L > 1$, so that as $\mathrm{d}_1, \ldots, \mathrm{d}_L \to \infty$, we can assume this recurrence exists:

   $$\begin{aligned} &\kappa^L(\mathbf{x}_\mathrm{i},\mathbf{x}_\mathrm{j}) \overset{(1)}{=} \tau^L(\mathbf{x}_\mathrm{i},\mathbf{x}_\mathrm{j}) + \kappa^{L-1}(\mathbf{x}_\mathrm{i},\mathbf{x}_\mathrm{j})\ddot{\boldsymbol{\sigma}}'(\tau^{L-1}(\mathbf{x}_\mathrm{i},\mathbf{x}_\mathrm{j})) \\ &\tau^L(\mathbf{x}_\mathrm{i},\mathbf{x}_\mathrm{j}) \overset{(2)}{=} \ddot{\boldsymbol{\sigma}}(\tau^{L-1}(\mathbf{x}_\mathrm{i},\mathbf{x}_\mathrm{j})) \\ &\tau^0(\mathbf{x}_\mathrm{i},\mathbf{x}_\mathrm{j}) = \mathbf{x}_\mathrm{i}{}^\intercal\mathbf{x}_\mathrm{j} \qquad \kappa^0(\mathbf{x}_\mathrm{i},\mathbf{x}_\mathrm{j}) = \ddot{\boldsymbol{\sigma}}'(\mathbf{x}_\mathrm{i}{}^\intercal\mathbf{x}_\mathrm{j}). \end{aligned} \quad \text{(E.42)}$$

Next we will prove equivalence (1) and (2) by deconstructing f layer-by-layer: by taking the derivatives of each layer l w.r.t all the terms in that layer, whilst holding the $\{l-1,\dots,1\}$ layers constant. Similar to E.33, the weight terms can be unrolled as:

$$\mathbf{w}^{L+1} = [\underbrace{w_{11}^{L+1},\dots,w_{\mathrm{d}_L 1}^{L+1}}_{\mathbf{W}^{L+1}},\ \underbrace{\underbrace{w_{11}^{L},\dots,w_{\mathrm{d}_L \mathrm{d}_{L-1}}^{L},}_{\mathbf{W}^{L}},\ \underbrace{w_{11}^{L-1},\dots\dots,\ w_{11}^{1},\dots,w_{\mathrm{d}_1\mathrm{d}_0}^{1}}_{\mathbf{w}^{L-1}=[\mathbf{W}^{L-1},\dots,\mathbf{W}^{1}]}}_{\mathbf{w}^{L}=[\mathbf{W}^{L},\dots,\mathbf{W}^{1}]}], \tag{E.43}$$

where $\mathbf{w} = \mathbf{w}^{L+1}$, and $\mathbf{w}^{L-1}$ is the set of weights without the terms from $\mathbf{W}^{L+1}$ and $\mathbf{W}^{L}$. Now similar to E.34, we can expand $f(\mathbf{w}^{L+1};\mathbf{x})$ into its terms and apply it to some $\mathbf{x} = (x_1,\dots,x_{\mathrm{d}_0})$. Dropping the normalization term, we can write then entire ANN as:

$$f(\mathbf{w}^{L+1};\mathbf{x}) \sim w_{11}^{L+1}\Big(\ddot{\boldsymbol{\sigma}}(w_{11}^{L}\ddot{\boldsymbol{\sigma}}(\dots\ddot{\boldsymbol{\sigma}}(w_{11}^{1}x_1+\dots+w_{1\mathrm{d}_0}^{1}x_{\mathrm{d}_0})))\Big)+\dots+w_{1\mathrm{d}_{L+1}}^{L+1}\Big(\ddot{\boldsymbol{\sigma}}(w_{11}^{L}\ddot{\boldsymbol{\sigma}}(\dots))\Big).$$

Now we can take the gradient w.r.t each term in E.43, and place them into a $1\times|\mathbf{w}|$ vector $\nabla f(\mathbf{w}^{L+1};\mathbf{x})$. The kernel at the $(L+1)$th layer is:

$$\begin{aligned}\kappa^{L}(\mathbf{x}_\mathsf{i},\mathbf{x}_\mathsf{j}) &= \Big\langle \nabla f(\mathbf{w}^{L+1};\mathbf{x}),\ \nabla f(\mathbf{w}^{L+1};\mathbf{x})\Big\rangle \\ &= \frac{1}{\mathrm{d}_L}\underbrace{\sum_{\mathsf{r}=1}^{\mathrm{d}_{L+1}}\nabla_{w_{1r}^{L+1}} f(\mathbf{w}^{L+1};\mathbf{x}_\mathsf{i})\cdot\nabla_{w_{1r}^{L+1}} f(\mathbf{w}^{L+1};\mathbf{x}_\mathsf{j})}_{\text{top layer}} + \underbrace{\Big\langle \nabla_{\mathbf{w}^{L}} f(\mathbf{w}^{L};\mathbf{x}_\mathsf{i}),\ \nabla_{\mathbf{w}^{L}} f(\mathbf{w}^{L};\mathbf{x}_\mathsf{j})\Big\rangle}_{\text{bottom } L \text{ layers}},\end{aligned}$$

where $f(\mathbf{w}^{L};\mathbf{x}_\mathsf{i}) = \mathbf{W}^{L+1}h^{L}(\mathbf{w}^{L};\mathbf{x})$. Now we consider the two terms and take the width of the layers to the limit.

a) *The top layer* is the neural network Gaussian process. Consider:

$$\begin{aligned}&\lim_{\mathrm{d}_L\to\infty}\frac{1}{\mathrm{d}_L}\sum_{\mathsf{r}=1}^{\mathrm{d}_{L+1}}\nabla_{w_{1r}^{L+1}} f(\mathbf{w}^{L+1};\mathbf{x}_\mathsf{i})\cdot\nabla_{w_{1r}^{L+1}} f(\mathbf{w}^{L+1};\mathbf{x}_\mathsf{j}) \\ &\qquad\overset{(1)}{=}\lim_{\mathrm{d}_L\to\infty}\frac{1}{\mathrm{d}_L}\sum_{\mathsf{r}=1}^{\mathrm{d}_{L}}\ddot{\boldsymbol{\sigma}}(\langle\mathbf{w}_r^{L},h^{L-1}(\mathbf{w}^{L-1};\mathbf{x}_\mathsf{i})\rangle)\cdot\ddot{\boldsymbol{\sigma}}(\langle\mathbf{w}_r^{L},h^{L-1}(\mathbf{w}^{L-1};\mathbf{x}_\mathsf{j})\rangle) \\ &\qquad\overset{(2)}{=}\mathbf{E}_{\mathbf{W}^{L}\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\Big[\Big\langle\ddot{\boldsymbol{\sigma}}(\mathbf{W}^{L}h^{L-1}(\mathbf{w}^{L-1};\mathbf{x}_\mathsf{i})),\ \ddot{\boldsymbol{\sigma}}(\mathbf{W}^{L}h^{L-1}(\mathbf{w}^{L-1};\mathbf{x}_\mathsf{j}))\Big\rangle\Big] \\ &\qquad\overset{(3)}{=}\mathbf{E}_{(u,v)\overset{\mathsf{iid}}{\sim}\mathcal{N}(\mathbf{0},\Lambda)}\Big[\Big\langle\ddot{\boldsymbol{\sigma}}(u),\ \ddot{\boldsymbol{\sigma}}(v)\Big\rangle\Big]\end{aligned}$$

$$\Lambda^{L-1} = \begin{bmatrix}\tau^{L-1}(\mathbf{x}_\mathsf{i},\mathbf{x}_\mathsf{i}) & \tau^{L-1}(\mathbf{x}_\mathsf{i},\mathbf{x}_\mathsf{j})\\ \tau^{L-1}(\mathbf{x}_\mathsf{j},\mathbf{x}_\mathsf{i}) & \tau^{L-1}(\mathbf{x}_\mathsf{j},\mathbf{x}_\mathsf{j})\end{bmatrix}\overset{(4)}{=}\begin{bmatrix}1 & \tau^{L-1}(\mathbf{x}_\mathsf{i},\mathbf{x}_\mathsf{j})\\ \tau^{L-1}(\mathbf{x}_\mathsf{j},\mathbf{x}_\mathsf{i}) & 1\end{bmatrix}.$$

Where going into (1), we take the gradient of $f(\mathbf{w};\mathbf{x}) = \mathbf{W}^{L+1}h^{L}(\mathbf{w};\mathbf{x})$ w.r.t its rth entry, so that every $s\neq r$ entry becomes zero. Going into (2), we use the law of large numbers and definition of inner product. Note that by assumption, the terms of $\mathbf{W}^{L}$ are drawn i.i.d via $w_{r1}^{L}\overset{\mathsf{iid}}{\sim}\mathcal{N}(0,1)$, so $\mathbf{W}^{L}\sim\mathcal{N}(\mathbf{0},\mathbf{I})$. Now by inductive hypothesis, $h^{L-1}(\mathbf{w};\mathbf{x}_\mathsf{i})$ is a Gaussian process, so that $\mathbf{W}^{L}\cdot h^{L-1}(\mathbf{w}^{L-1};\mathbf{x}_\mathsf{i})$ is a product of two Gaussian distributed vectors, therefore it is also Gaussian distributed. Thus by a similar argument to E.23, we take step (3): by setting $u = \mathbf{W}^{L}\cdot h^{L-1}(\mathbf{w}^{L-1};\mathbf{x}_\mathsf{i})$, and $v = \mathbf{W}^{L}\cdot h^{L-1}(\mathbf{w}^{L-1};\mathbf{x}_\mathsf{j})$, we can construct a new Gaussian distribution with mean $\mathbf{0}$ and covariance Λ^{L-1} over vector u,v. Going

into (4), we first use the inductive hypothesis to recall in the base case this is true: $\tau^0(\mathbf{x}_\mathsf{i}, \mathbf{x}_\mathsf{j}) = \mathbf{x}_\mathsf{i}{}^\intercal \mathbf{x}_\mathsf{j}$. Then note that in dual activation, $\ddot{\boldsymbol{\sigma}}(\mathbf{x}_\mathsf{i}{}^\intercal \mathbf{x}_\mathsf{i}) = 1$ if $\mathbf{x}_\mathsf{i}$ is drawn from a Gaussian distribution with support on the surface of a sphere. Thus we have $\tau^{L-1}(\mathbf{x}_\mathsf{i}, \mathbf{x}_\mathsf{i}) = 1$. Thus we have the recurrence over vectors (u, v):

$$\tau^L(\mathbf{x}_\mathsf{i}, \mathbf{x}_\mathsf{j}) = \mathbf{E}_{(u,v)\overset{\text{iid}}{\sim}\mathcal{N}(\mathbf{0},\Lambda)}\left[\left\langle \ddot{\boldsymbol{\sigma}}(u),\ \ddot{\boldsymbol{\sigma}}(v)\right\rangle\right] = \ddot{\boldsymbol{\sigma}}(\tau^{L-1}(\mathbf{x}_\mathsf{i}, \mathbf{x}_\mathsf{j})),$$

which is what equality two of E.42 states.

b) *The "bottom L layers:"* consider the gradient over the hidden layers $\nabla_{\mathbf{w}^L} f(\mathbf{w}^L; \mathbf{x}_\mathsf{i})$. In this case, we generalize to any layer $1 \leq l \leq L-1$. Now fix two indices within some $w^l_{\mathsf{st}} = (\mathbf{W}^l)_{st}$: $s \in \{1, \ldots, \mathrm{d}_{l+1}\}$, and $t \in \{1, \ldots, \mathrm{d}_l\}$. Now consider the gradient of the ANN w.r.t the values at coordinate st:

$$\begin{aligned}\nabla_{w^l_{\mathsf{st}}}\left(\mathbf{W}^{L+1}h^L(\mathbf{w};\mathbf{x})\right) &\overset{(1)}{=} \frac{1}{\sqrt{\mathrm{d}_L}}\sum_{\mathsf{r}=1}^{\mathrm{d}_L} \mathbf{w}^{L+1}_{r1}\ddot{\boldsymbol{\sigma}}'((\mathbf{W}^L h^{L-1}(\mathbf{w}^{L-1};\mathbf{x}))_\mathsf{r}) \\ &\quad \cdot \left(\nabla_{w^l_{\mathsf{st}}}(h^{L-1}(\mathbf{w}^{L-1};\mathbf{x}))_\mathsf{r}\right).\end{aligned}$$

Where going into (1), we expand the product of $\mathbf{W}^{L+1}$ with $h^L(\mathbf{w};\mathbf{x})$ by the rows of $\mathbf{W}^{L+1} = [\mathbf{w}^{L+1}_{11}, \ldots, \mathbf{w}^{L+1}_{\mathrm{d}_L 1}]$. And then apply the chain rule to differentiate the activation function, followed by pushing the differential to the terms inside the differentiated $\ddot{\boldsymbol{\sigma}}'(\ldots)$. Now consider the inner product of the gradients of the above term:

$$\begin{aligned}&\lim_{\mathrm{d}_L\to\infty}\left\langle \nabla_{w^l_{\mathsf{st}}}(\mathbf{W}^{L+1}h^L(\mathbf{w};\mathbf{x}_\mathsf{i})),\ \nabla_{w^l_{\mathsf{st}}}(\mathbf{W}^{L+1}h^L)(\mathbf{w};\mathbf{x}_\mathsf{j})\right\rangle \\ &\quad\overset{(1)}{=} \lim_{\mathrm{d}_L\to\infty}\frac{1}{\mathrm{d}_L}\sum_{r,r'=1}^{\mathrm{d}_L} \mathbf{w}^{L+1}_{r1}\ddot{\boldsymbol{\sigma}}'((\mathbf{W}^L h^{L-1}(\mathbf{w}^{L-1};\mathbf{x}_\mathsf{i}))_\mathsf{r}) \\ &\qquad\qquad \cdot \mathbf{w}^{L+1}_{r'1}\ddot{\boldsymbol{\sigma}}'((\mathbf{W}^L h^{L-1}(\mathbf{w}^{L-1};\mathbf{x}_\mathsf{j}))'_r) \\ &\qquad\qquad \cdot \left\langle \nabla_{w^l_{\mathsf{st}}} h^{L-1}(\mathbf{w}^{L-1};\mathbf{x}_\mathsf{i}),\ \nabla_{w^l_{\mathsf{st}}} h^{L-1}(\mathbf{w}^{L-1};\mathbf{x}_\mathsf{j})\right\rangle \\ &\quad\overset{(2)}{=} \lim_{\mathrm{d}_L\to\infty}\frac{1}{\mathrm{d}_L}\sum_{\mathsf{r}=1}^{\mathrm{d}_L} (\mathbf{w}^{l+1}_{r1})^2 \cdot \ddot{\boldsymbol{\sigma}}'((\mathbf{W}^L h^{L-1}(\mathbf{w}^{L-1};\mathbf{x}_\mathsf{j}))_\mathsf{r}) \\ &\qquad\qquad \cdot \ddot{\boldsymbol{\sigma}}'((\mathbf{W}^L h^{L-1}(\mathbf{w}^{L-1};\mathbf{x}_\mathsf{j}))_\mathsf{r}) \cdot \kappa^{L-1}(\mathbf{x}_\mathsf{i}, \mathbf{x}_\mathsf{j}) \\ &\quad\overset{(3)}{=} \lim_{\mathrm{d}_L\to\infty}\frac{1}{\mathrm{d}_L}\sum_{\mathsf{r}=1}^{\mathrm{d}_L} \ddot{\boldsymbol{\sigma}}'((\mathbf{W}^L h^{L-1}(\mathbf{w}^{L-1};\mathbf{x}_\mathsf{j}))_\mathsf{r}) \cdot \ddot{\boldsymbol{\sigma}}'((\mathbf{W}^L h^{L-1}(\mathbf{w}^{L-1};\mathbf{x}_\mathsf{j}))_\mathsf{r}) \cdot \kappa^{L-1}(\mathbf{x}_\mathsf{i}, \mathbf{x}_\mathsf{j}) \\ &\quad\overset{(4)}{=} \ddot{\boldsymbol{\sigma}}'(\tau^{L-1}(\mathbf{x}_\mathsf{i}, \mathbf{x}_\mathsf{j})) \cdot \kappa^{L-1}(\mathbf{x}_\mathsf{i}, \mathbf{x}_\mathsf{j}).\end{aligned}$$

Here going into (1), we apply the expanded definition of $\nabla_{w^l_{\mathsf{st}}}\left(\mathbf{W}^{L+1}h^L(\mathbf{w};\mathbf{x})\right)$ from above. The indices of the sum range over r, r' where when $r = r'$, the inner product is $\kappa^{L-1}(\mathbf{x}_\mathsf{i}, \mathbf{x}_\mathsf{j})$, and zero when $r \neq r'$. Going into (2), note that $\langle \nabla_{w^l_{\mathsf{st}}} h^{L-1}(\mathbf{w}^{L-1};\mathbf{x}_\mathsf{i}), \nabla_{w^l_{\mathsf{st}}} h^{L-1}(\mathbf{w}^{L-1};\mathbf{x}_\mathsf{j})\rangle$ is the definition of kernel at layer $L-1$: $\tau^{L-1}(\mathbf{x}_\mathsf{i}, \mathbf{x}_\mathsf{j})$. Going into (3), we note that in expectation $(\mathbf{w}^{L+1}_{r1})^2 = 1$, as the terms of $\mathbf{W}^{L+1}$ are drawn i.i.d from $\mathcal{N}(0, 1)$. Going into (4), we use the definition of the kernel again.

Finally we have:

$$\kappa^L(\mathbf{x}_\mathsf{i}, \mathbf{x}_\mathsf{j}) = \tau^L(\mathbf{x}_\mathsf{i}, \mathbf{x}_\mathsf{j}) + \kappa^{L-1}(\mathbf{x}_\mathsf{i}, \mathbf{x}_\mathsf{j})\ddot{\boldsymbol{\sigma}}'(\tau^{L-1}(\mathbf{x}_\mathsf{i}, \mathbf{x}_\mathsf{j})) \qquad \kappa^0(\mathbf{x}_\mathsf{i}, \mathbf{x}_\mathsf{j}) = 0$$

as desired [[LB+18], [AD+19], [AD+20]].

■

APPENDIX F

Exponential Family Distributions

This section provides a bare minimum introduction to the exponential family, how it is constructed, and the basic characteristics of its partition function. The question here is how to price the difference in quality between two models of the exponential family, and how various representations of exponential models can be used in said pricing.

F.1 Derivation

In the motivating setting, suppose there is a random vector $\mathbf{x} = \{\mathsf{x}_1, .., \mathsf{x}_\mathsf{m}\}$ taking value over sample space Ω^m, with distribution $\mathfrak{p}$. Now we wish to express the distribution $\mathfrak{p}$ using a finite set of parameters. The natural step is to compute the *moments* of the distribution,

$$\hat{\mu}_\alpha = \varphi_\alpha(\mathbf{x}) := \frac{1}{\mathsf{n}} \sum_{\mathsf{s}=1}^{\mathsf{m}} \mathsf{x}_\mathsf{s}^\alpha \qquad (1 \leq \alpha \leq d).$$

The value $\hat{\mu}_\alpha$ is read "the αth degree empirical mean of the dataset". The φ_α are the *sufficient statistics* of the distribution of degree α, each one maps realizations of some r.v. onto reals, $\varphi_\alpha : \Omega^\mathsf{m} \to \mathbb{R}$. For example $\varphi_2(\mathbf{x}) = \frac{1}{\mathsf{n}} \sum_{\mathsf{s}=1}^{\mathsf{m}} \mathsf{x}_\mathsf{s}^2$, and note the sufficient statistics in the first and second degree are the sample mean and variance of the dataset. Now we wish to select the probability distribution $\mathfrak{p}$ that generates the sample means *in expectation*, namely this condition must be satisfied:

$$\mathbf{E}_\mathfrak{p}[\varphi_\alpha(\mathbf{x})] := \int_{\Omega^\mathsf{m}} \varphi_\alpha(\mathbf{x}) \mathfrak{p}(\varphi_\alpha(\mathbf{x}))\ \nu(d\mathbf{x}) = \hat{\mu}_\alpha \quad (1 \leq \alpha \leq d).$$

That is to say the expected value of the sufficient statistics under every possible sample of $\mathbf{x}$ is the same as the sample mean we observe from the current selection. This set of constraints is called the *moment matching conditions* of selecting the distribution $\mathfrak{p}$. It turns out, there are many distributions $\mathfrak{p}$ that lies within the boundaries defined by the moment-matching constraints, and the principle of minimum entropy is used to select the best $\mathfrak{p}$. Recall entropy is $H(\mathfrak{p}) = -\int \mathfrak{p}(\mathbf{x}) \log(\mathfrak{p}(\mathbf{x}))\nu(d\mathbf{x})$.

So the complete optimization problem solves for:

$$\mathfrak{p}^* = \arg\max_{\mathfrak{p} \in \mathfrak{P}} H(\mathfrak{p}) \quad \text{subject to } \mathbf{E}_\mathfrak{p}[\varphi_\alpha(\mathbf{x})] = \hat{\mu}_\alpha \quad (1 \leq \alpha \leq d).$$

Solving this program involves finding the minimum point $\mathfrak{p}^*$ on the level sets of $H(\mathfrak{p})$ that is incident to the curves traced out by the moment matching constraints. At the solution point, the gradient of $H(\mathfrak{p})$ is equal to the gradient of the constraints up to some constant factor - that is we have $\nabla H(\mathfrak{p}) = \theta_\alpha \nabla(\mathbf{E}_\mathfrak{p}[\varphi_\alpha(\mathbf{x})] = \hat{\mu}_\alpha)$ for every α. The θ_α's are Lagrange multipliers (although typically the symbol λ is used) that parameterize the distribution $\mathfrak{p}$ alongside the sufficient statistics [BV04]. In the context of exponential family, the θ's are also called *canonical parameters*. Solving for $\mathfrak{p}^*$, we find the distribution of maximum entropy that agrees with the data is of form $\exp(\langle \boldsymbol{\theta}, \boldsymbol{\varphi}(\mathbf{x}) \rangle)$, with the proper normalization factor omitted here for clarity.

F.2 Canonical Representation

Formally, let the random vector $\mathbf{x} = (\mathsf{x}_1, .., \mathsf{x}_\mathsf{s}, .., \mathsf{x}_\mathsf{m})$ take value in the Cartesian product space $\Omega^\mathsf{m} = \Omega_1 \times ... \times \Omega_\mathsf{m}$. Ω may be discrete, ie $\{0, 1\}$ or it may be continuous. Ω^m is endowed with measure ν defined as $d\nu = h(\mathbf{x})dx$, where $h \colon \Omega^\mathsf{m} \to \mathbb{R}$. In the discrete case, $\nu(d\mathbf{x})$ is simply the counting measure, in the continuous case it may be the Lebesgue measure. Now define the sufficient statistic $\varphi_\alpha \colon \Omega^\mathsf{m} \to \mathbb{R}$ as $\varphi_\alpha(\mathbf{x}) = \frac{1}{\mathsf{n}} \sum_\mathsf{s}^\mathsf{m} \mathsf{x}_\mathsf{s}^\alpha$. And define $\boldsymbol{\varphi} \colon \Omega^\mathsf{m} \to \mathbb{R}^d$ as $\boldsymbol{\varphi} = (\varphi_1, .., \varphi_d)$, a vector of functions that computes the sufficient statistics of values $\mathbf{x} \in \Omega^\mathsf{m}$ up to the dth degree. A statistic $\varphi(\mathbf{x})$ is sufficient w.r.t parameter $\boldsymbol{\theta}$, if $\mathfrak{p}(\mathbf{x}|\boldsymbol{\varphi}(\mathbf{x}); \boldsymbol{\theta})$ does not depend on any value of this statistic $\boldsymbol{\varphi}(\mathbf{x})$. Next associate $\boldsymbol{\varphi}$ with the canonical parameters $\boldsymbol{\theta} = (\theta_1, .., \theta_d)$, then the *exponential family* is a collection of density functions:

$$\mathfrak{p}\big(\mathbf{x}; \boldsymbol{\theta}\big) = \exp\Big(\langle \boldsymbol{\varphi}(\mathbf{x}), \boldsymbol{\theta} \rangle - \mathsf{A}(\boldsymbol{\theta}) \Big).$$

Where the normalization factor, or *log partition function* $A(\theta)$ is:

$$\mathsf{A}(\boldsymbol{\theta}) = \log \int_{\Omega^\mathsf{m}} \exp\Big(\langle \boldsymbol{\varphi}(\mathbf{x}), \boldsymbol{\theta} \rangle \Big) \, \nu(d\mathbf{x}).$$

This quantity is of particular interest in our case as it is the cost function. The function A has some important properties w.r.t its first and second derivatives that will be useful in the model market, namely the derivatives of A are the moments of the density, and A is convex.

Theorem F.2.1. *The log partition function has derivatives of all orders on its domain Θ, and the αth derivative of the partition function is the αth centered-moment of the sufficient statistic. In particular, the first two derivatives yield the cumulants of the random vector $\boldsymbol{\varphi}(\mathbf{x})$ with:*

$$\frac{\partial A}{\partial \theta_\alpha}(\boldsymbol{\theta}) = \mathbf{E}_{\mathfrak{p}(\cdot;\boldsymbol{\theta})}[\varphi_\alpha(\mathbf{x})] = \int_{\Omega^m} \varphi_\alpha(\mathbf{x}) \mathfrak{p}(\mathbf{x}; \boldsymbol{\theta}) \nu(d\mathbf{x}). \tag{F.1}$$

$$\frac{\partial^2 A}{\partial \theta_\alpha \theta_\beta}(\boldsymbol{\theta}) = \mathbf{E}_{\mathfrak{p}(\cdot;\boldsymbol{\theta})}[\varphi_\alpha(\mathbf{x})\varphi_\beta(\mathbf{x})] - \mathbf{E}_{\mathfrak{p}(\cdot;\boldsymbol{\theta})}[\varphi_\alpha(\mathbf{x})]\mathbf{E}_{\mathfrak{p}(\cdot;\boldsymbol{\theta})}[\varphi_\beta(\mathbf{x})]. \tag{F.2}$$

Note the expectations are taken with respect to probability distribution parameterized by $\boldsymbol{\theta}$. We then have the Jacobian:

$$\nabla \mathsf{A}(\boldsymbol{\theta}) = \left(\frac{\partial A}{\partial \theta_\alpha}(\boldsymbol{\theta}) \, : \, 1 \leq \alpha \leq d \right), \tag{F.3}$$

and the Hessian matrix:

$$\nabla^2 \mathsf{A}(\boldsymbol{\theta}) = \left[\frac{\partial^2 A}{\partial \theta_\alpha \theta_\beta}(\boldsymbol{\theta}) \, : \, (1,1) \leq (\alpha, \beta) \leq (d, d) \right]. \tag{F.4}$$

Note the Hessian specifies the variance of the distribution. And because the variance is always > 0, A must be convex in every θ. Now define the set Θ:

$$\Theta = \{ \boldsymbol{\theta} \in \mathbb{R}^d \, : \, \mathsf{A}(\theta) < \infty \},$$

note Θ must be a convex set as A is a convex function. Θ is strictly convex if the representation of A is minimal, that is to say the density is not over-parameterized with more sufficient statistics φ_α than necessary [[WJ04], [WJ08]].

Example F.2.2 (Ising Model)**.** The Ising model is a kind of Markov Random Field (MRF), whereby the graph $G = (V, E)$ has Bernoulli random variables $\mathsf{x_s}$ associated with every vertex $\mathsf{s} \in V$. The edges on the graph is associated with weight θ_{st} for $\mathsf{s}, t \in V$, and every vertex is associated with potential θ_{s}. The density for this model is then:

$$\mathfrak{p}(\mathbf{x}; \boldsymbol{\theta}) = \exp\Big(\sum_{\mathsf{s}\in V} \theta_{\mathsf{s}}\mathsf{x_s} + \sum_{(\mathsf{s},\mathsf{t})\in E} \theta_{\mathsf{st}}\mathsf{x_s}\mathsf{x_t} - \mathsf{A}(\boldsymbol{\theta})\Big).$$

Note the canonical parameter characterize the graph with $\boldsymbol{\theta} = (\theta_1, .., \theta_{|V\cup E|})$ [WJ04].

F.3 Mean Representation

It is useful to define an alternate formulation of the exponential family using the *mean parameters* $\boldsymbol{\mu}$, with $A(\boldsymbol{\mu})$ as the partition function. Translating between the two representations will be of interest because although $A(\boldsymbol{\theta})$ is used to price the market updates, it is very difficult to evaluate in practice.

Since $A(\boldsymbol{\mu})$ is defined over the mean parameters, it is useful to name the set:

$$\mathcal{M} = \Big\{\boldsymbol{\mu} \in \mathbb{R}^d : \int_{\Omega^{\mathsf{m}}} \boldsymbol{\varphi}(\mathbf{x})\mathfrak{p}(\mathbf{x})\nu(d\mathbf{x}) = \boldsymbol{\mu} \text{ for some } \mathfrak{p} \in \mathfrak{Q}\Big\},$$

and notice the set $\mathfrak{Q}$ may include distributions that are not in the exponential family. However the probability distribution that maximizes entropy is from the exponential family. In the case where the r.v. X is discrete, we have:

$$\mathcal{M} = \Big\{\boldsymbol{\mu} \in \mathbb{R}^d : \sum_{\mathbf{x}\in\Omega^{\mathsf{m}}} \boldsymbol{\varphi}(\mathbf{x})\mathfrak{p}(\mathbf{x}) = \boldsymbol{\mu} \text{ for some } \mathfrak{p}(\mathbf{x}) > 0, \sum_{\mathbf{x}\in\Omega^{\mathsf{m}}} \mathfrak{p}(\mathbf{x}) = 1\Big\}.$$

That is the mean parameter set is just the convex hull of the sufficient statistics, this follows from the fact that the expression $\sum_{\mathbf{x}\in\Omega^{\mathsf{m}}} \boldsymbol{\varphi}(\mathbf{x})\mathfrak{p}(\mathbf{x})$ is the convex combination of $\boldsymbol{\varphi}(\mathbf{x})$'s.

Example F.3.1 (Gaussian Markov Random Field)**.** Returning to the MRF, the mean parameters of MRF are the mean vector $\boldsymbol{\mu} = \mathbf{E}[X]$ and the covariance $\Sigma = \mathbf{E}[XX^{\intercal}]$. Thus its mean parameter set is:

$$\mathcal{M} = \Big\{(\boldsymbol{\mu}, \Sigma) : \Sigma - \boldsymbol{\mu}^{\intercal}\boldsymbol{\mu} \succeq 0\Big\}.$$

And the density is:

$$\mathfrak{p}(\mathbf{x}; \boldsymbol{\mu}, \Sigma) = h(\mathbf{x}) \exp\Big(-\frac{1}{2}tr\big(\Sigma^{-1}\mathbf{x}\mathbf{x}^{\intercal}\big) + \boldsymbol{\mu}^{\intercal}\Sigma^{-1}\mathbf{x} - \frac{1}{2}\mathsf{A}(\boldsymbol{\mu}, \Sigma)\Big),$$

where $A(\boldsymbol{\mu}, \Sigma)$ and $h(\mathbf{x})$ are normalization factors. This is the more familiar representation of the Gaussian [WJ04].

F.4 Kullback-Leibler Divergence

KL divergence measures the distance between two distributions, this quantity will be used to price the contribution of successive updates on the underlying statistical model. There are many ways to introduce KL divergence, in this context it is illuminating to express it w.r.t. $A(\boldsymbol{\theta})$, and the hyperplanes that supports the epigraph of A at various $\boldsymbol{\theta}$'s along the curve. Referencing the figure $F.1$, there are $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$, which parametrize $\mathfrak{p}(\cdot; \boldsymbol{\theta}_1)$ and $\mathfrak{p}(\cdot; \boldsymbol{\theta}_2)$ respectively. Now suppose we want to find the "distance" between the two $\boldsymbol{\theta}$'s. The simplest way to do so is to integrate along the curve $A(\boldsymbol{\theta})$, but this would be a prohibitive

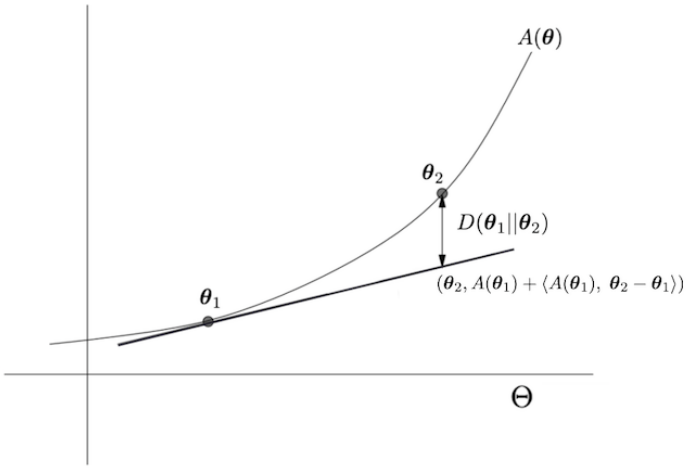


Figure F.1: The KL divergence of two distributions parameterized by $\boldsymbol{\theta}$ is the gap between the hyperplane supporting $\boldsymbol{\theta}_1$ evaluated at the point $\boldsymbol{\theta}_2$, and the curve $A(\boldsymbol{\theta}_2)$. Observe if we fix $\boldsymbol{\theta}_1$ and move $\boldsymbol{\theta}_2$ so that it is close to right side of where $A(\boldsymbol{\theta})$ is defined, then the KL divergence grows very large [WJ04].

task depending on the form of A. Instead, we do a first order approximation by drawing a hyperplane from $\boldsymbol{\theta}_1$ with slope $\nabla \mathsf{A}(\boldsymbol{\theta}_1)$. The plane intersects θ_2 at the value:

$$\mathsf{A}(\boldsymbol{\theta}_1) + \Big\langle \nabla \mathsf{A}(\boldsymbol{\theta}_1),\ \boldsymbol{\theta}_2 - \boldsymbol{\theta}_1 \Big\rangle.$$

And we see there is a gap between the actual value of $A(\boldsymbol{\theta}_2)$ and the value of the expression above (see fig (F.1)). This gap is the KL divergence, written:

$$\begin{aligned}
\mathsf{D}(\boldsymbol{\theta}_1||\boldsymbol{\theta}_2) &= \mathsf{A}(\boldsymbol{\theta}_2) - \Big(\mathsf{A}(\boldsymbol{\theta}_1) + \Big\langle \nabla \mathsf{A}(\boldsymbol{\theta}_1),\ \boldsymbol{\theta}_2 - \boldsymbol{\theta}_1 \Big\rangle\Big) \\
&= \mathsf{A}(\boldsymbol{\theta}_2) - \mathsf{A}(\boldsymbol{\theta}_1) - \Big\langle \nabla \mathsf{A}(\boldsymbol{\theta}_1),\ \boldsymbol{\theta}_2 - \boldsymbol{\theta}_1 \Big\rangle \\
&= \mathsf{A}(\boldsymbol{\theta}_2) - \mathsf{A}(\boldsymbol{\theta}_1) - \Big\langle \boldsymbol{\mu}_1,\ \boldsymbol{\theta}_2 - \boldsymbol{\theta}_1 \Big\rangle. \qquad \text{(F.5)}
\end{aligned}$$

Where the last line follows from the fact that by the moment matching condition, we have $\nabla \mathsf{A}(\boldsymbol{\theta}_1) = \boldsymbol{\mu}_1$. This form is called the *primal form* of the KL divergence, or the *Bregman Divergence*. Alternatively we can arrive at the expression above starting from the more familiar definition of KL divergence:

$$\mathsf{D}(\mathfrak{q}||\mathfrak{p}) = \int_{\Omega^m} \mathfrak{q}(\mathbf{x}) \log \frac{\mathfrak{q}(\mathbf{x})}{\mathfrak{p}(\mathbf{x})} \nu(d\mathbf{x}) = \mathbf{E}_{\mathfrak{q}}\Big[\log \frac{\mathfrak{q}(\mathbf{x})}{\mathfrak{p}(\mathbf{x})}\Big]. \qquad \text{(F.6)}$$

Applying the definition of $\mathfrak{p}$, we have:

$$\begin{aligned}
\mathsf{D}(\mathfrak{p}(\mathbf{x};\boldsymbol{\theta}_1)||\mathfrak{p}(\mathbf{x};\boldsymbol{\theta}_2)) &= \mathbf{E}_{\mathfrak{p}(\cdot;\boldsymbol{\theta}_1)}\Big[\log \frac{\mathfrak{p}(\mathbf{x};\boldsymbol{\theta}_1)}{\mathfrak{p}(\mathbf{x};\boldsymbol{\theta}_2)}\Big] \\
&= \mathbf{E}_{\mathfrak{p}(\cdot;\boldsymbol{\theta}_1)}\Big[\log(\mathfrak{p}(\mathbf{x};\boldsymbol{\theta}_1)) - \log(\mathfrak{p}(\mathbf{x};\boldsymbol{\theta}_2))\Big] \\
&= \mathbf{E}_{\mathfrak{p}(\cdot;\boldsymbol{\theta}_1)}\Big[\Big(\langle \boldsymbol{\varphi}(\mathbf{x}), \boldsymbol{\theta}_1\rangle - \mathsf{A}(\boldsymbol{\theta}_1)\Big) - \Big(\langle \boldsymbol{\varphi}(\mathbf{x}), \boldsymbol{\theta}_2\rangle - \mathsf{A}(\boldsymbol{\theta}_2)\Big)\Big]
\end{aligned}$$

$$
\begin{aligned}
&= \mathbf{E}_{\mathfrak{p}(\cdot;\boldsymbol{\theta}_1)}\Big[\Big(\langle\boldsymbol{\varphi}(\mathbf{x}),\boldsymbol{\theta}_1\rangle - \langle\boldsymbol{\varphi}(\mathbf{x}),\boldsymbol{\theta}_2\rangle\Big) - \Big(\mathsf{A}(\boldsymbol{\theta}_1) - \mathsf{A}(\boldsymbol{\theta}_2)\Big)\Big] \\
&= \int_{\Omega^{\mathsf{m}}} \mathfrak{p}(\mathbf{x};\boldsymbol{\theta}_1)\Big(\Big\langle \boldsymbol{\varphi}(\mathbf{x}),\ \boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\Big\rangle - (\mathsf{A}(\boldsymbol{\theta}_1) - \mathsf{A}(\boldsymbol{\theta}_2))\Big)\nu(d\mathbf{x}) \\
&= \mathsf{A}(\boldsymbol{\theta}_2) - \mathsf{A}(\boldsymbol{\theta}_1) + (\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2)\int_{\Omega^{\mathsf{m}}} \boldsymbol{\varphi}(\mathbf{x})\mathfrak{p}(\mathbf{x};\boldsymbol{\theta}_1)\nu(d\mathbf{x}) \\
&= \mathsf{A}(\boldsymbol{\theta}_2) - \mathsf{A}(\boldsymbol{\theta}_1) - \langle\boldsymbol{\mu}_1,\boldsymbol{\theta}_2 - \boldsymbol{\theta}_1\rangle.
\end{aligned}
$$

Where going to last line, we use the fact that the expected value of the sufficient statistics under density $\mathfrak{p}(\cdot;\boldsymbol{\theta}_1)$ is $\boldsymbol{\mu}_1$ by the moment matching constraints. And thus the two expressions of KL divergence are in correspondence. Alternatively we can express the divergence in terms of its dual form:

$$
\mathsf{D}(\boldsymbol{\mu}_1||\boldsymbol{\mu}_2) = \mathsf{A}^*(\boldsymbol{\mu}_1) - \mathsf{A}^*(\boldsymbol{\mu}_2) - \langle\boldsymbol{\theta}_1,\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2\rangle,
$$

and note the signs are flipped because $A(\boldsymbol{\theta})$ is dual with $-\mathsf{A}^*(\boldsymbol{\mu})$.

We can also start from $\boldsymbol{\theta}_2$ and find its distance to $\boldsymbol{\theta}_1$:

$$
\begin{aligned}
&\mathsf{A}(\boldsymbol{\theta}_2) + \langle\nabla\mathsf{A}(\boldsymbol{\theta}_2),\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\rangle && \text{(hyperplane from } \boldsymbol{\theta}_2\text{)} \\
&\mathsf{D}(\boldsymbol{\theta}_2||\boldsymbol{\theta}_1) = \mathsf{A}(\boldsymbol{\theta}_1) - (\mathsf{A}(\boldsymbol{\theta}_2) + \langle\nabla\mathsf{A}(\boldsymbol{\theta}_2),\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\rangle) && \text{(KL divergence from } \boldsymbol{\theta}_2 \text{ to } \boldsymbol{\theta}_1\text{)}.
\end{aligned}
$$

Notice $\mathsf{D}(\boldsymbol{\theta}_1||\boldsymbol{\theta}_2) \neq \mathsf{D}(\boldsymbol{\theta}_2||\boldsymbol{\theta}_1)$, so KL divergence is not a symmetric distance function, this is clear from the picture: the curve has different gradients at different points. Similarly, we can define the KL divergence for the conditional distribution. In this case, a set of random variables is partitioned into observed variables y and unobserved variables $\mathbf{x}$, and the probability model is a joint exponential family distribution for $(\mathbf{x},\mathsf{y})$) [WJ04]:

$$
\mathfrak{p}(\mathbf{x}|\mathsf{y};\boldsymbol{\theta}) = \frac{\exp(\langle\boldsymbol{\theta},\boldsymbol{\varphi}(\mathbf{x},\mathsf{y})\rangle)}{\int_{\Omega^{\mathsf{m}}}\exp(\langle\boldsymbol{\theta},\boldsymbol{\varphi}(\mathbf{x},\mathsf{y})\rangle)\nu(d\mathbf{x})} = \exp\Big(\langle\boldsymbol{\theta},\boldsymbol{\varphi}(\mathbf{x},\mathsf{y})\rangle + \mathsf{A}_y(\boldsymbol{\theta})\Big).
$$

The Bregman divergence of conditional distribution is then comparable to eqn (F.5):

$$
\begin{aligned}
&\mathsf{D}(\mathfrak{p}(\mathbf{x}|\mathsf{y};\boldsymbol{\theta}_1)||\mathfrak{p}(\mathbf{x}|\mathsf{y};\boldsymbol{\theta}_2)) = \mathbf{E}_{\mathfrak{p}(\mathbf{x}|\mathsf{y};\boldsymbol{\theta}_1)}\Big[\log(\mathfrak{p}(\mathbf{x}|\mathsf{y};\boldsymbol{\theta}_1)) - \log(\mathfrak{p}(\mathbf{x}|\mathsf{y};\boldsymbol{\theta}_2))\Big] \\
&= \int_{dom(Y)}\mathfrak{p}(\mathsf{y})\int_{\Omega^{\mathsf{m}}}\mathfrak{p}(\mathbf{x}|\mathsf{y};\boldsymbol{\theta}_1)\Big(\langle\boldsymbol{\varphi}(\mathbf{x},\mathsf{y}),\boldsymbol{\theta}_1\rangle - \mathsf{A}_y(\boldsymbol{\theta}_1) - \langle\boldsymbol{\varphi}(\mathbf{x},\mathsf{y}),\boldsymbol{\theta}_2\rangle + \mathsf{A}_y(\boldsymbol{\theta}_2)\Big)\nu(d\mathbf{x})\nu(dy) \\
&= \mathsf{A}_y(\boldsymbol{\theta}_2) - \mathsf{A}_y(\boldsymbol{\theta}_1) + (\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2)\int_{\mathsf{y}\in dom(Y)}\mathfrak{p}(\mathsf{y})\int_{\Omega^{\mathsf{m}}}\mathfrak{p}(\mathbf{x}|\mathsf{y};\boldsymbol{\theta}_1)\boldsymbol{\varphi}(\mathbf{x},\mathsf{y})\nu(d\mathbf{x})\nu(dy) \\
&= \mathsf{A}_y(\boldsymbol{\theta}_2) - \mathsf{A}_y(\boldsymbol{\theta}_1) - \langle\boldsymbol{\mu}_1,\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\rangle. \qquad \text{(F.7)}
\end{aligned}
$$

Where going into the last line, we used the moment matching condition to set

$$
\int_{\mathsf{y}\in dom(Y)}\mathfrak{p}(\mathsf{y})\int_{\Omega^{\mathsf{m}}}\mathfrak{p}(\mathbf{x}|\mathsf{y};\boldsymbol{\theta}_1)\boldsymbol{\varphi}(\mathbf{x},\mathsf{y})\nu(d\mathbf{x})\nu dy = \mathbf{E}_{\mathfrak{p}(\cdot|\mathsf{y};\boldsymbol{\theta}_1)}[\boldsymbol{\varphi}(\mathbf{x},\mathsf{y})] = \nabla\mathsf{A}_y(\boldsymbol{\theta}_1) = \boldsymbol{\mu}_1.
$$

In more familiar form, we have:

$$
\begin{aligned}
\mathsf{D}(\mathfrak{p}(\mathbf{x}|\mathsf{y};\boldsymbol{\theta}_1)||\mathfrak{p}(\mathbf{x}|\mathsf{y};\boldsymbol{\theta}_2)) &= \mathbf{E}_{\mathfrak{p}(\mathbf{x}|\mathsf{y};\boldsymbol{\theta}_1)}\Big[\mathsf{D}(\mathfrak{p}(\mathbf{x}|\mathsf{y};\boldsymbol{\theta}_1)||\mathfrak{p}(\mathbf{x}|\mathsf{y};\boldsymbol{\theta}_2))\Big] \\
&= \int_{dom(Y)}\mathfrak{p}(\mathsf{y})\Big(\int_{\Omega^{\mathsf{m}}}\mathfrak{p}(\mathbf{x}|\mathsf{y};\boldsymbol{\theta}_1)\cdot\log\frac{\mathfrak{p}(\mathbf{x}|\mathsf{y};\boldsymbol{\theta}_1)}{\mathfrak{p}(\mathbf{x}|\mathsf{y};\boldsymbol{\theta}_2)}\nu(d\mathbf{x})\Big)\nu(d\mathsf{y}). \qquad \text{(F.8)}
\end{aligned}
$$

Its discrete analogue is:

$$
\mathsf{D}(\mathfrak{p}(\mathbf{x}|\mathsf{y};\boldsymbol{\theta}_1)||\mathfrak{p}(\mathbf{x}|\mathsf{y};\boldsymbol{\theta}_2)) = \sum_{\mathsf{y}_{\mathsf{s}}\in dom(Y)}\mathfrak{p}(\mathsf{y}_{\mathsf{s}})\cdot\sum_{(\mathbf{x}_{\mathsf{s}},\mathsf{y}_{\mathsf{s}})}\mathfrak{p}(\mathbf{x}_{\mathsf{s}}|y_{\mathsf{s}};\boldsymbol{\theta}_1)\log\frac{\mathfrak{p}(\mathbf{x}_{\mathsf{s}}|\mathsf{y}_{\mathsf{s}};\boldsymbol{\theta}_1)}{\mathfrak{p}(\mathbf{x}_{\mathsf{s}}|\mathsf{y}_{\mathsf{s}};\boldsymbol{\theta}_2)}.
$$

This form can be computed via a great number of of sampling methods.

Bibliography

[ACV12] Abernethy, J., Chen, Y. and Vaughan, J. W. 'Efficient Market Making via Convex Optimization, and a Connection to Online Learning'. In: *ACM Transactions on Economics and Computation* (2012).

[AD+19] Arora, S., Du, S. et al. 'On Exact Computation with an Infinitely Wide Neural Net'. In: *Neural Information Processing Systems* (2019).

[AD+20] Arora, S., Du, S. et al. 'Harnessing the Power of Infinitely Wide Deep Nets on Small-data Tasks'. In: *ICLR* (2020).

[Agr+09] Agrawal, S. et al. 'A Unified Framework for Dynamic Pari-Mutuel Information Market Design'. In: *Proceedings of the 10th ACM conference on Electronic commerce* (2009).

[AK+14] Abernethy, J., Kutty, S. et al. 'Information Aggregation in Exponential Family Markets'. In: *Proceedings of the fifteenth ACM conference on Economics and computation* (2014).

[AM+23] A., K., M., G. et al. 'Will the real stablecoin please stand up?' In: *BIS Papers* (2023).

[AS22] A., K.-M. and S., S. 'Designing Autonomous Markets for Stablecoin Monetary Policy'. In: *arxiv* (2022).

[AW+21] Alemohammad, S., Wang, Z. et al. 'The Recurrent Neural Tangent Kernel'. In: *ICRL* (2021).

[Axl20] Axler, S. *Measure, Integration and Real Analysis*. Springer, 2020.

[Bar08] Bartlett, P. 'Representer theorem and kernel examples'. 2008.

[Bea73] Beals, R. *Advanced Mathematical Analysis: Periodic Functions and Distributions, Complex Analysis, Laplace Transform and Applications*. Springer, 1973.

[Bel21] Belkin, M. 'Fit without fear: remarkable mathematical phenomena of deep learning through the prism of interpolation'. In: *Cambridge University Press* (2021).

[BMR21] Bartlett, P., Montanari, A. and Rakhlin, A. 'Deep learning: a statistical viewpoint'. In: *Cambridge University Press* (2021).

[BP98] Brin, S. and Page, L. 'The anatomy of a large-scale hypertextual Web search engine'. In: *Computer Networks* (1998).

[BV04] Boyd, S. and Vandenberghe, L. *Convex Optimization*. Cambridge University Press, 2004.

[CA10] Chen, Y. and Abernethy, J. 'An Optimization-Based Framework for Automated Market-Making'. In: *12th ACM Conference on Electronic Commerce* (2010).

[CL06] Cesa-Bianchi, N. and Lugosi, G. *Prediction, Learning, and Games*. Cambridge University Press, 2006.

[CM99] Cho, G. and Meyer, C. 'Markov chain sensitivity measured by mean first passage times'. In: *Linear Algebra and its Applications* (1999).

[CP07] Chen, Y. and Pennock, D. 'A Utility Framework for Bounded-Loss Market Makers'. In: *Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence* (2007).

[CV10] Chen, Y. and Vaughan, J. W. 'A New Understanding of Prediction Markets via No-Regret Learning'. In: *11th ACM Conference on Electronic Commerce* (2010).

[D25] D., C. 'Petrodollar to Digital Yuan'. In: *asiasociety* (2025).

[Dom20] Domingos, P. 'Every Model Learned by Gradient Descent Is Approximately a Kernel Machine'. In: *arxiv* (2020).

[DR14] Dwork, C. and Roth, A. *The Algorithmic Foundations of Differential Privacy*. Foundations and Trends in Theoretical Computer Science, 2014.

[FBS20] Farina, G., Bianchi, T. and Sandholm1, T. 'Coarse Correlation in Extensive-Form Games'. In: *AAAI* (2020).

[For+07] Fortnow, C. et al. 'Betting on Permutations'. In: *EC* (2007).

[FPW23] Frongillo, R., Papireddygari, M. and Waggoner, B. 'An Axiomatic Characterization of CFMMs and Equivalence to Prediction Markets'. In: *arXiv* (2023).

[GB16] Goodfellow, I. and Bengio, Y. *Deep Learning*. MIT Press, 2016.

[GH+24] Globus-Harris, I., Harrison, D. et al. 'Diversified Ensembling: An Experiment in Crowdsourced Machine Learning'. In: *FAccT* (2024).

[GKD19] Görtler, J., Kehlbeck, R. and Deussen, O. 'A Visual Exploration of Gaussian Processes'. 2019.

[GP+14] Goodfellow, I., Pouget-Abadie, J. et al. 'Generative Adversarial Nets'. In: *Advances in neural information processing systems (pp. 2672–2680)* (2014).

[GR07] Gneiting, T. and Raftery, A. 'Strictly Proper Scoring Rules, Prediction, and Estimation'. In: *Journal of American Statistical Association* (2007).

[Han02] Hanson, R. 'Logarithmic Market Scoring Rules for Modular Combinatorial Information Aggregation'. In: *The Journal of Prediction Markets* (2002).

[Han03] Hanson, R. 'Combinatorial Information Market Design'. In: *Information Systems Frontiers* (2003).

[Har+20] Harz, A. K.-M. et al. 'Stablecoins 2.0: Economic Foundations and Risk-based Models'. In: *ACM* (2020).

[HB+20] Hron, J., Bahri, Y. et al. 'Infinite attention: NNGP and NTK for deep attention networks'. In: *Proceedings of Machine Learning Research* (2020).

[Hil24] Hill, E. 'Artificial Bubbles:The Rise and Fall of "AI"'. In: *The Observer Vol 48.* (2024).

[hkm22] hkma. 'mBridge: Building a multi CBDC platform for international payments'. In: *hkma* (2022).

[HSW89] Hornik, K., Stinchcombe, M. and White, H. 'Multilayer feedforward networks are universal approximators'. In: *Neural Networks* (1989).

[Hug24] HugginFace. 'huggingface.co/models'. In: (2024).

[HW19] Harris, J. and Waggoner, B. 'Decentralized and Collaborative AI on Blockchain'. In: *IEEE* (2019).

[IW06] Ipsen, C. and Wills, R. 'Mathematical Properties and Analysis of Google's PageRank'. In: *Bol. Soc. Esp. Mat.* (2006).

[JG+18] Jacot, A., Gabriel, F. et al. 'Neural Tangent Kernel: Convergence and Generalization in Neural Networks'. In: *NIPS'18: Proceedings of the 32nd International Conference on Neural Information Processing Systems* (2018).

[JM24] Jurafsky, D. and Martin, J. *Speech and Language Processing (3rd ed. draft)*. 2024.

[KP+23] Kwon, Y., Pongmala, K. et al. 'What Drives the (In)stability of a Stablecoin?' In: *arxiv* (2023).

[Kut15] Kutty, S. 'all men count with you, but none too much: Information Aggregation and Learning in Prediction Markets'. In: (2015).

[Lal18] Lalley, S. *Random Walks on Infinite Discrete Groups*. Northwestern, 2018.

[LB+18] Lee, J., Bahri, Y. et al. 'Deep Neural Networks as Gaussian Processes'. In: *ICRL* (2018).

[LB18] Lee, J. and Bahri, Y. 'Deep Neural Networks as Gaussian Processes'. In: *neurips* (2018).

[LM98] Langville, A. and Meyer, C. 'Deeper Inside PageRank'. In: *Internet Mathematics Vol. 1, No. 3: 335-380* (1998).

[LPW17] Levin, D., Peres, Y. and Wilmer, E. *Markov Chains and Mixing Times, second edition*. AMS, 2017.

[LX+19] Lee, J., Xiao, L. et al. 'Wide Neural Networks of Any Depth Evolve as Linear Models Under Gradient Descent'. In: *neurips* (2019).

[LZB21] Liu, C., Zhu, L. and Belkin, M. 'On the linearity of large non-linear models: when and why the tangent kernel is constant'. In: *neurips* (2021).

[Mat20] Matheau-Raven, O. *Random Walks deon the Symmetric Group: Cutoff for One-sided Transposition Shuffles*. arXiv, 2020.

[MC+13] Mikolov, T., Chen, K. et al. 'Efficient Estimation of Word Representations in Vector Space'. In: *arxiv* (2013).

[Mit13] Mitzenmacher, M. 'Tossing a Biased Coin'. In: (2013).

[MP+24] Metaxa, ., Park, J. et al. *Auditing Algorithms: Understanding Algorithmic Systems from the Outside In*. Human-Computer Interaction: Vol. 14, No. 4, 2024.

[MR95] Motwani, R. and Raghavan, P. *Randomized Algorithms*. Cambridge University Press, 1995.

[Nea96] Neal, R. *Bayesian Learning for Neural Networks*. Springer-Verlag New York, Inc, 1996.

[Nic19] Nichani, E. 'Random Walks on Groups'. In: (2019).

[ODo21] O'Donnell, R. *Analysis of Boolean Functions*. arXiv, 2021.

[Pen04] Pennock, D. 'A Dynamic Pari-Mutuel Market for Hedging, Wagering, and Information Aggregation'. In: *Proceedings of the 5th ACM conference on Electronic commerce* (2004).

[Pow23] Powell, W. *China, Trust and Digital Supply Chains*. Routledge, 2023.

[Rad+22a] Radhakrishnan, A. et al. 'Kernel Regression'. 2022.

[Rad+22b] Radhakrishnan, A. et al. 'NNGP, Dual Activations, and Over-Parameterization'. 2022.

[Rad+22c] Radhakrishnan, A. et al. 'NTK for Deep Networks and Convolutional NTK (CNTK)'. 2022.

[Rad+22d] Radhakrishnan, A. et al. 'NTK Origin and Derivation'. 2022.

[Rak09] Rakhlin, A. 'Lecture Notes on Online Learning'. 2009.

[Rou14] Roughgarden, T. *Lectures Notes on Algorithmic Game Theory*. Tim Roughgarden, 2014.

[RW06] Rasmussen, C. and Williams, I. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[SC04] Shawe-Taylor, J. and Cristianini, N. *Kernel Methods for Pattern Analysis*. The Edinburgh Building, Cambridge CB2 2RU, UK: cambridge university press, 2004.

[Sch00] Scholkopf, B. 'The Kernel Trick for Distances'. In: *Neural Information Processing Systems* (2000).

[SCM21] Salehi, M., Clark, J. and Mannan, M. 'Red-Black Coins: Dai Without Liquidations'. In: *Financial Cryptography and Data Security* (2021).

[Sha48] Shannon. 'A Mathematical Theory of Communication'. In: *The Bell System Technical Journal* (1948).

[Shr10] Shreve, S. *Stochastic Calculus for Finance II: Continuous-Time Models*. Springer, 2010.

[SM+23] S., W., M., B. et al. 'Central Bank Digital Currency Global Interoperability Principles'. In: *The World Economic Forum* (2023).

[SS+17] Silver, D., Schrittwieser, J. et al. 'Mastering the game of Go without human knowledge'. In: *Nature* (2017).

[SS16] Saitoh, S. and Sawano, Y. *Theory of Reproducing Kernels and its Applications*. Springer, 2016.

[SW22] Seabrook, E. and Wiskott, L. 'A Tutorial on the Spectral Theory of Markov Chains'. In: *arxiv* (2022).

[Tad13] Tadelis, S. *Game Theory: An Introduction*. Princeton University Press, 2013.

[Tan23] Tanaka, Y. 'On the Average Hitting Times of Weighted Caley Graphs'. In: *arxiv* (2023).

[TJ04] Thibaux, R. and Jordan, M. 'CS281B/Stat241B: Advanced Topics in Learning and Decision Making'. 2004.

[UBS21] UBS. 'Project mBridge'. In: *UBS* (2021).

[Van23] Vanschoren, J. 'Lecture 3: Kernelization'. 2023.

[VS+17] Vaswani, A., Shazeer, N. et al. 'Attention Is All You Need'. In: *neurips* (2017).

[Was19] Wasserman, L. 'Function Spaces'. 2019.

[Wel19] Welling, M. 'Kernel ridge Regression'. 2019.

[WFA15] Waggoner, B., Frongillo, R. and Abernethy, J. 'A Market Framework for Eliciting Private Data'. In: *NIPS* (2015).

[WJ04] Wainwright, M. and Jordan, M. *Graphical Models, Exponential Families, and Variational Inference*. Berkeley 94720, CA: Foundations and Trends in Machine Learning, 2004.

[WJ08] Wainwright, M. and Jordan, M. *Graphical Models, Exponential Families, and Variational Inference*. Berkeley 94720, CA: Foundations and Trends in Machine Learning, 2008.

[Yan20] Yang, G. 'Tensor Programs II: Neural Tangent Kernel for Any Architecture'. In: *arxiv* (2020).

[YBS22] Yuan, J., Barmpoutis, P. and Stathaki, T. 'Effectiveness of Vision Transformer for Fast and Accurate Single-Stage Pedestrian Detection'. In: *neurips* (2022).

[YL20] Yang, G. and Littwin, E. 'Tensor Programs IIb: Architectural Universality of Neural Tangent Kernel Training Dynamics'. In: *Proceedings of Machine Learning Research* (2020).

[ZJ+12] Zhang, Jin et al. 'Efficient Online Learning for Large-Scale Sparse Kernel Logistic Regression'. In: *AAAI* (2012).