Contents

1. Preliminaries

Let the current best ask price at time t be level 0, and let Y be a random variable representing the future best ask price at time $t + \Delta t$. For simplicity, assume the price at level k is simply the integer k.

Claim 1. *If $y > 0$ is a level of the limit order book at time t, then conditional on $Y \geqslant y$, the probability that $Y > y$ strongly depends on the size x_y of the limit order book at y.*

This claim is not explicitly proven in the paper. However, strong statistical evidence is given in support of it. Furthermore, we make an assertion about the direction of the dependency.

Claim 2. *The conditional probability that $Y > y$ given that $Y \geqslant y$ decreases with x_y, the size at level y.*

The intuition given in the paper is as follows:

> "To reach a level $y' > y$ the sell limit orders at level y must be consumed by buy orders. The larger the ask size at level y. the less likely the future best ask price will reach a level $y' > y$."

The author further mentions that "the behavior of the best ask price as it moves upwards is analogous to a 'geometric random variable' whose probability of increasing from y to $y+1$ depends on the size x_y at level y."

Thus, we take a moment to give context and a formal definition for a geometric random variable.

Definition 3. The **geometric distribution** is the probability distribution of the number X of Bernoulli trials needed to get one success, supported on the set $\{1, 2, 3, \dots\}$.

Definition 4. Given that the assumptions for modeling with a geometric distribution are satisfied, the **geometric random variable** X is the total number of trials up to and including the first success, and the number of failures preceding the first success is $X - 1$.

In this case, a success is defined as a move from level y to some level $y' > y$, and the variable X is the number of time steps we need to traverse to get a first success.

Question 5. *What is the intuition for the probability $\mathbb{P}[Y > y | Y \geqslant y]$, assuming $y > 0$?*

First, what does it mean to assume $y > 0$? Since 0 is the level of the best ask price at time t, $y > 0$ assumes we are considering some ask price above the current best. Next, what does the condition $Y \geqslant y$ mean? This means we are restricting ourselves to the case where the future best ask price Y at time $t + \Delta t$ is at least y, which means we are ruling out the case in which the future ask price does not increase, and we are also ruling out all cases where it increases by less than y levels. And finally, the event we are judging the probability of is $Y > y$, so we are asking the following.

> *What is the probability that the future best ask price increases by more than y levels if we know it increases by at least y levels, where y is some positive integer?*

Question 6. *Why do we condition on $Y \geqslant y$? Is it not possible for the current best ask price to drop from time t to time $t + \Delta t$? How do we model the probability of this occurring?*

4. Neural Network Architectures for Modeling Distributions on $\mathbb{R}^d$

Understanding Section 4.1 is key. Let $\mathcal{X}$ be the input space of the model in question, and let $\mathcal{Y}$ be a finite, discrete output space. Furthermore, let $f_{\theta,L} : \mathcal{X} \to \mathcal{Y}$ be a neural network, where L is the number of layers, and $\theta = (W_1, \dots, W_L, b_1, \dots, b_L)$ are the parameters (weight matrices and bias terms). Let d_l be the dimension of layer l, such that $W_l \in \mathbb{R}^{d_l} \times \mathbb{R}^{d_{l-1}}$, and $d_L = |\mathcal{Y}|$. Note also that the final layer of $f_{\theta,L}$ is the softmax function:

$$g(z) = \left(\frac{e^{z_1}}{\sum_{i=1}^{d_L} e^{z_i}}, \dots, \frac{e^{z_{d_L}}}{\sum_{i=1}^{d_L} e^{z_i}} \right), z \in \mathbb{R}^{d_L}. \tag{1}$$

4.1. **Straightforward application to allow generalization.**

Remark 7. The term **unnormalized log probability** should really read *log of the unnormalized probability* (see unnormalized-log-prob). So the unnormalized log probability under $f_{\Theta,L}$ from the Section 4.1 would be $\log g_{\Theta,L}(x,y) = \log f_{\Theta,L}(x)_{k_y}$, where $1 \leqslant k_y \leqslant d_L$, i.e. k_y is the index of the relevant component corresponding to the event y of the output space $\mathcal{Y}$, or, equivalently, the relevant component of the output vector given by $f_{\Theta,L}$, which lives in $\mathbb{R}^{d_L}$. Recall y is one of $d_L = |\mathcal{Y}|$ finite events which can be computed.

Thus, in Section 4.2, the author states that the probability of y given x in the above setting is given by

$$\frac{e^{f_\theta(x,y)}}{\sum_{y' \in \mathcal{Y}} e^{f_\theta(x,y')}}. \tag{2}$$

Question 8. *How is this meant to be implemented? Are we supposed to take an argmax over each of the $|\mathcal{Y}|$ probabilities for each input x and then optimize that to predict the true event $\hat{y}$ corresponding to the input x?*

4.3. **A computationally efficient architecture for modeling spatial distributions.**

4.3.1. *Extension to $\mathcal{R}^d$.* Let $\mathcal{R}^d$ be the d-dimensional lattice where $\mathcal{R} = \dots, r_{-2}, r_{-1}, 0, r_1, r_2, \dots$. We follow Example 4.2 from the paper. Let Y_1 be a random variable representing the change in best ask price, and let Y_2 be a random variable representing the change in best bid price. We measure change in levels as in Section 1. Thus we have $Y_1 \in \mathbb{Z}$, and $Y_2 \in \mathbb{Z}$, and thus $(Y_1, Y_2) \in \mathbb{Z}^2$.

We now follow the constructions in Section 4.3.1, adapting to the $d = 2$ case above. Let $Y = (Y_1, Y_2) \in \mathcal{R}^2$ have the conditional distribution:

$$\begin{aligned} \mathbb{P}[Y = (y_1, y_2) | X = x] &= \mathbb{P}[Y_1 = y_1 | X = x] \mathbb{P}[Y_2 = y_2 | Y_{0:1} = y_{0:1}, X = x]; \\ \mathbb{P}[Y_1 = y_1] &= g_\theta^1(x, y_1); \\ \mathbb{P}[Y_2 = y_2 | Y_{0:1} = y_{0:1}, X = x] &= g_\theta^2(x, y_{0:1}, y_2). \end{aligned} \tag{3}$$

The conditional distributions g^1, g^2 will be functions of neural networks, which will be specified shortly.

Let $f_\theta^{1,-} : \mathcal{X} \times \mathbb{R} \to \mathbb{R}$ and $f_\theta^{1,+} : \mathcal{X} \times \mathbb{R} \to \mathbb{R}$ be neural networks.

The conditional distribution of Y_1 conditional on X is completely specified by

$$\begin{cases} \mathbb{P}[Y_1 = y_1 | Y_1 \geqslant y_1, X = x] = \frac{e^{f_\theta^{1,+}(x,y_1)}}{1+e^{f_\theta^{1,+}(x,y_1)}} & y_1 \geqslant r_1 \\ \mathbb{P}[Y_1 = z | X = x] = h_\theta^{1,z}(x) & z \in \{ y_1 > 0, y_1 = 0, y_1 < 0 \} . \\ \mathbb{P}[Y_1 = y_1 | Y_1 \leqslant y_1, X = x] = \frac{e^{f_\theta^{1,-}(x,y_1)}}{1+e^{f_\theta^{1,-}(x,y_1)}} & y_1 \leqslant r_1 \end{cases} \tag{4}$$

But note the above expression does not give us a formula for $g_\theta^1(x, y_1)$, we must derive it, and it is recursively defined by the above formulas.

The neural network $h_\theta^1(x)$ is a standard neural network for classification which produces a vector of three probabilities for the events $y_1 > 0, y_1 = 0, y_1 < 0$, and $h_\theta^{1,z}(x)$ is the z-th vector element of $h_\theta^1(x)$, so $z \in \{ 1, 2, 3 \}$.

Let's consider for a moment only the *increase* network, given by

$$\mathbb{P}[Y_1 = y_1 | Y_1 \geqslant y_1, X = x] = \frac{e^{f_\theta^{1,+}(x,y_1)}}{1 + e^{f_\theta^{1,+}(x,y_1)}} \quad y_1 \geqslant r_1. \tag{5}$$

Note that since $\mathcal{R} \cong \mathbb{Z}$ is discrete, the statement $y_1 \geqslant r_1$ is equivalent to saying $y_1 > 0$. Now how would we implement training of the network $f_\theta^{1,+}$? Note x is the input, which is a sequence of truncated order books with adjusted volumes. Then y_1 is a price level. It is **not** the next order book, the target, the ground truth for the next level change, or anything analogous. It is more like a query. We are saying:

Given this input x, what is the probability that the ask price at time $t + \Delta t$ is equal to this query y_1, assuming that the future ask price is at least y_1, i.e. it increases by at least y_1 levels, i.e. what is $\mathbb{P}[Y_1 = y_1 | Y_1 \geqslant y_1, X = x]$?

Note this implicitly gives us the probability that the future ask price is greater than y_1 as well, since we are conditioning on $Y_1 \geqslant y_1$ (just subtract the computed probability from 1 to get the probability of an increase to a level greater than y_1).

Also note that in previous sections we did not treat the case allowing for decreases in best ask price, but that is handled here by the third line in 12.

There are a couple of key differences between Sirignano's setting and ours. The first is that he seems to assume a fixed-size order book, whereas we do not, and thus his claim that it is feasible to model the entire order book via neural networks is not true for a cheeky reason in our case, namely that close to the two ends of the order book, the number of levels is variable, and thus the models may have no inputs to go on for an arbitrary number of time steps, and we also may not spin up enough models to deal with the maximum number of levels seen at evaluation time. This is easily fixed by simply limiting the number of models to be far below the lowest number of levels seen in training data. Assuming the number of levels on both sides of the order book at evaluation time is a random sample drawn from the same distribution as the order books from the training set, we can construct a confidence interval for the minimum (or first quartile or some other statistic) number of levels in the order book, and then choose our number of models on each side such that we are sufficiently confident it will not exceed the number of visible levels at any point. This would only be necessary in the case where we wanted to model as much of the order book as possible, which very likely isn't true. In fact, minimizing the number of models is very valuable for computational complexity reasons, and it may not even be prudent to model for more than a single level on each side ($d = 2$), i.e. the best bid and best ask.

The second, and more worrisome difference is that Sirignano's setting assumes that the price levels are discrete and atomic. We know from examination of our data that this is not actually true (there are far too many missing levels at the true atomic price difference size to model). Thus we use adjusted volumes for a dense subset of the order book, where the prices all differ by the atomic, smallest price difference. The problem with this approach is that it does not take into account the case where levels spontaneously arise between existing levels, i.e. a price for which there was previously 0 volume on a given side of the order book now has nonzero volume, and this change has taken place simultaneously with a prediction made by the model.

A correction: it appears that Sirignano does not assume the nonzero levels are dense. He allows for and acknowledges the existence of many levels with zero volume.

Question 9. *Is it useful to know how many levels the price changes? Or do we only need the output of the ternary classification model?*

Question 10. *How often does the best bid/ask price change?*

Question 11. *How often does the best bid/ask price move deeper into the order book?*

Question 12. *How often does the best bid/ask price add a level above/below the best bid/ask price (shallower), i.e. how often does the spread become smaller?*

Question 13. *How often does the best bid/ask price change to a level that previously did not exist with nonzero volume in the order book?*

Question 14. *How often does the best bid/ask price change by more than one level?*

Question 15. *What does the distribution of differences between levels with nonzero sizes look like close to the best bid/ask prices?*

Since the set of y_i values chosen to compute the output of either the increase or decrease network, i.e. $\mathbb{P}[Y_i = y_i | Y_i = y_i, X = x_i]$, determines how much information about the predicted distribution we know, it may be a good idea to choose these strategically, instead of a wide net of all possible values around the best bid/ask prices. This is especially true if the differences between nonzero values are large. We could try varying the y_i values chosen time step to time step. A heuristic for determining which to compute could be as follows: compute the probability for all y_i which have nonzero volume in the relevant region at time step t. Then compute a uniformly spaced grid of values where the grid spacing is determined by the distribution of the magnitudes of nonzero level differences. The grid could also have spacing which gets exponentially larger as the level gets higher, to reflect the lower probability that the best bid/ask price reaches that level, and thus the lesser importance of that information.

Question 16. *I need to understand Equation 11. I need to ask Craw about this.*

Recall the formula for **cross entropy** given for discrete probability distributions p and q with the same support $\mathcal{Y}$, which is

$$H_{p,q}(x) = -\sum_{y \in \mathcal{Y}} p(x, y) \log q(x, y). \tag{6}$$

In the context of machine learning and optimization, the expression $\log q(x, y)$ is the log likelihood of the output y for the input x, and $p(x, y)$ is the true label.

Remark 17. The log likelihood function $\mathcal{L}(\{ (x, y) \})$ given in Equation 13 in the paper is exactly this $\log q(x, y)$.

Question 18. *What is the support $\mathcal{Y}$ in the context of Equations 11 and 13 in the paper?*

It is the same $\mathcal{Y}$ as in the paper. This is what the distribution is *over*.
Read this: cross-entropy.

We would like to determine the true labels $p(x, y)$. What is the true label? It is the true probability that $Y = y$ given x. Let y' be the true best bid/ask price at time $t + \Delta t$. Then $p(x, y') = 1$ and $p(x, y) = 0$ for all $y \neq y'$.

Question 19. *How is the loss function usually computed for large pretrained transformer models? Is it a classification loss? Can we replicate it in this setting? Can we use negative sampling?*

These models use cross-entropy.

Remark 20. The problem that I am seeing is that there is an ordering to the y values in the order book case, and that is not true in the vocabulary/transformer case. So we don't want to just blindly penalize the model for guessing because we want the model to know how close its guess was.

Remark 21. Note that Equation 13 in the paper is just the natural log of the first line of Equation 12.

We give an expanded formula for the cross-entropy of a single training example $x \in \mathcal{X}$, where $y = (y_1, y_2)$:

$$\begin{aligned} H_{p,q}(x) &= -\sum_{y \in \mathcal{Y}} p(x,y) \log q(x,y) \\ &= -\sum_{y \in \mathcal{Y}} p(x,y) \mathcal{L}(\{\,(x,y)\,\}) \\ &= -\sum_{y \in \mathcal{Y}} p(x,y) \log\left(\mathbb{P}[Y = (y_1, y_2) | X = x]\right) \\ &= -\sum_{y \in \mathcal{Y}} p(x,y) \log\left(\mathbb{P}[Y_1 = y_1 | X = x]\mathbb{P}[Y_2 = y_2 | Y_{0:1} = y_{0:1}, X = x]\right) \\ &= -\sum_{y \in \mathcal{Y}} p(x,y) \left[\log\left(g_\theta^1(x, y_1)\right) + \log\left(g_\theta^2(x, y_{0:1}, y_2)\right)\right] \end{aligned} \tag{7}$$

Question 22. *How do we compute $g_\theta^1(x, y_1)$? Recall that $g_\theta^1(x, y_1)$ is just shorthand for the conditional distribution $\mathbb{P}[Y_1 = y_1 | X = x]$.*

Let $y_1' = y_1 - 1$. Then can we compute $\mathbb{P}[Y_1 \geqslant y_1 | X = x]$ from $\mathbb{P}[Y_1 = y_1' | Y_1 \geqslant y_1', X = x]$? Well we know

$$\mathbb{P}[Y_1 > y_1' | Y_1 \geqslant y_1', X = x] = 1 - \mathbb{P}[Y_1 = y_1' | Y_1 \geqslant y_1', X = x]. \tag{8}$$

And we know $Y_1 > y_1'$ if and only if $Y_1 \geqslant y_1$ since y_1' is the largest level smaller than y_1 (levels are discrete). Thus this will be a recursive definition. Let $r_x : \mathbb{N} \to [0, 1]$ be given by

$$r_x(j) = \mathbb{P}[Y_1 = j | Y_1 \geqslant j, X = x]. \tag{9}$$

We omit the condition $X = x$ for brevity (it still applies for all probability expressions in what follows). Then the expanded form of this definition in terms of $r_x(j)$ is

$$\begin{aligned}
\mathbb{P}[Y_1 = y_1] &= \mathbb{P}[Y_1 = y_1 | Y_1 \geqslant y_1]\mathbb{P}[Y_1 \geqslant y_1] \\
&= r_x(y_1)\mathbb{P}[Y_1 > y_1 - 1] \\
&= r_x(y_1)\mathbb{P}[Y_1 > y_1 - 1 | Y_1 \geqslant y_1 - 1]\mathbb{P}[Y_1 \geqslant y_1 - 1] \\
&= r_x(y_1)[1 - r_x(y_1 - 1)]\mathbb{P}[Y_1 \geqslant y_1 - 1] \\
&= r_x(y_1)[1 - r_x(y_1 - 1)]\mathbb{P}[Y_1 > y_1 - 2] \\
&= r_x(y_1)[1 - r_x(y_1 - 1)]\mathbb{P}[Y_1 > y_1 - 2 | Y_1 \geqslant y_1 - 2]\mathbb{P}[Y_1 \geqslant y_1 - 2] \\
&= r_x(y_1)[1 - r_x(y_1 - 1)][1 - r_x(y_1 - 2)]\mathbb{P}[Y_1 \geqslant y_1 - 2] \qquad (10) \\
&\vdots \\
&= r_x(y_1) \prod_{j=1}^{y_1 - 1} [1 - r_x(j)]\, \mathbb{P}[Y_1 \geqslant 1] \\
&= \mathbb{P}[Y_1 = y_1 | Y_1 \geqslant y_1] \prod_{j=1}^{y_1 - 1} [1 - \mathbb{P}[Y_1 = j | Y_1 \geqslant j]]\, \mathbb{P}[Y_1 \geqslant 1].
\end{aligned}$$

The seventh line is the same operation as was done on the third line, we now write $\mathbb{P}[Y_1 > y_1 - 2 | Y_1 \geqslant y_1 - 2, X = x]$ as $1 - r_x(y_1 - 2)$, and continue, yielding the product formula in the second-to-last line.

Recall that the definition of conditional probability gives us that $\mathbb{P}[Y_1 = y_1, Y_1 \geqslant y_1] = \mathbb{P}[Y_1 = y_1 | Y_1 \geqslant y_1]\mathbb{P}[Y_1 \geqslant y_1]$, but since $Y_1 = y_1 \Rightarrow Y_1 \geqslant y_1$, we have that $\mathbb{P}[Y_1 = y_1, Y_1 \geqslant y_1] = \mathbb{P}[Y_1 = y_1]$. Note that we we use $\mathbb{P}[A, B]$ to denote $\mathbb{P}[A \cap B]$ throughout. Thus we have the first line of Equation 10. Also note that the same logic applies when we write $\mathbb{P}[Y_1 > j] = \mathbb{P}[Y_1 > j | Y_1 \geqslant j]\mathbb{P}[Y_1 \geqslant j]$, since $\mathbb{P}[Y_1 > j, Y_1 \geqslant j] = \mathbb{P}[Y_1 > j]$, for $j \in \mathbb{N}$.

Adding back in our $X = x$ conditions, the final formula is

$$g_\theta^1(x, y_1) = \mathbb{P}[Y_1 = y_1 | Y_1 \geqslant y_1, X = x] \prod_{j=1}^{y_1 - 1} [1 - \mathbb{P}[Y_1 = j | Y_1 \geqslant j, X = x]]\, \mathbb{P}[Y_1 \geqslant 1 | X = x]. \qquad (11)$$

But note that $\mathbb{P}[Y_1 \geqslant 1 | X = x] = \mathbb{P}[Y_1 > 0 | X = x]$ because $\mathcal{Y}$ is discrete. And $\mathbb{P}[Y_1 > 0 | X = x]$ is given to us by $h_\theta^{1,z}(x)$, so we simply multiply by the output of the neural network h. Recall

The conditional distribution of Y_1 conditional on X is completely specified by

$$\begin{cases}
\mathbb{P}[Y_1 = y_1 | Y_1 \geqslant y_1, X = x] = \frac{e^{f_\theta^{1,+}(x, y_1)}}{1 + e^{f_\theta^{1,+}(x, y_1)}} & y_1 \geqslant r_1 \\
\mathbb{P}[Y_1 = z | X = x] = h_\theta^{1,z}(x) & z \in \{ y_1 > 0, y_1 = 0, y_1 < 0 \}\,. \\
\mathbb{P}[Y_1 = y_1 | Y_1 \leqslant y_1, X = x] = \frac{e^{f_\theta^{1,-}(x, y_1)}}{1 + e^{f_\theta^{1,-}(x, y_1)}} & y_1 \leqslant r_1
\end{cases} \qquad (12)$$

Hence, substituting in expressions involving f_θ^1 and h_θ^1 for the conditional distributions, we know that for $y_1 > 0$,

$$\begin{aligned}
g_\theta^1(x, y_1) &= \frac{e^{f_\theta^{1,+}(x, y_1)}}{1 + e^{f_\theta^{1,+}(x, y_1)}} \prod_{j=1}^{y_1 - 1} \left[1 - \frac{e^{f_\theta^{1,+}(x, j)}}{1 + e^{f_\theta^{1,+}(x, j)}} \right] \mathbb{P}[Y_1 > 0 | X = x] \\
&= \frac{e^{f_\theta^{1,+}(x, y_1)}}{1 + e^{f_\theta^{1,+}(x, y_1)}} \prod_{j=1}^{y_1 - 1} \left[1 - \frac{e^{f_\theta^{1,+}(x, j)}}{1 + e^{f_\theta^{1,+}(x, j)}} \right] h_\theta^{1, y_1 > 0}(x).
\end{aligned} \qquad (13)$$

And we can write the entire expression for g_θ^1 in all cases ($y_1 > 0, y_1 = 0, y_1 < 0$) as a piecewise function:

$$g_\theta^1(x, y_1) = \begin{cases} \dfrac{e^{f_\theta^{1,+}(x,y_1)}}{1 + e^{f_\theta^{1,+}(x,y_1)}} \displaystyle\prod_{j=1}^{y_1-1} \left[1 - \dfrac{e^{f_\theta^{1,+}(x,j)}}{1 + e^{f_\theta^{1,+}(x,j)}}\right] h_\theta^{1,y_1>0}(x) & \text{if } y_1 > 0 \\ h_\theta^{1,y_1=0}(x) & \text{if } y_1 = 0 \\ \dfrac{e^{f_\theta^{1,-}(x,y_1)}}{1 + e^{f_\theta^{1,-}(x,y_1)}} \displaystyle\prod_{j=1}^{y_1-1} \left[1 - \dfrac{e^{f_\theta^{1,-}(x,j)}}{1 + e^{f_\theta^{1,-}(x,j)}}\right] h_\theta^{1,y_1<0}(x) & \text{if } y_1 < 0 \end{cases}. \tag{14}$$

Question 23. *But what about g_θ^2?*

Ignoring the confusing index notation, we simply include y_1 as an input to $f_\theta^{2,+}$, $f_\theta^{2,-}$, and $h_\theta^{2,z}$. Thus

$$g_\theta^2(x, y_1, y_2) = \begin{cases} \dfrac{e^{f_\theta^{2,+}(x,y_1,y_2)}}{1 + e^{f_\theta^{2,+}(x,y_1,y_2)}} \displaystyle\prod_{j=1}^{y_2-1} \left[1 - \dfrac{e^{f_\theta^{2,+}(x,y_1,j)}}{1 + e^{f_\theta^{2,+}(x,y_1,j)}}\right] h_\theta^{2,y_2>0}(x, y_1) & \text{if } y_2 > 0 \\ h_\theta^{2,y_2=0}(x, y_1) & \text{if } y_2 = 0 \\ \dfrac{e^{f_\theta^{2,-}(x,y_1,y_2)}}{1 + e^{f_\theta^{2,-}(x,y_1,y_2)}} \displaystyle\prod_{j=1}^{y_2-1} \left[1 - \dfrac{e^{f_\theta^{2,-}(x,y_1,j)}}{1 + e^{f_\theta^{2,-}(x,y_1,j)}}\right] h_\theta^{2,y_2<0}(x, y_1) & \text{if } y_2 < 0 \end{cases}. \tag{15}$$

Question 24. *Recall that $g_\theta^2(x, y_1, y_2)$ is just shorthand for $\mathbb{P}[Y_2 = y_2 | Y_1 = y_1, X = x]$. How many values of y_1 and y_2 do we have to compute to take the argmax over the entire support for y_1 and y_2, i.e. to decide which values of y_1 and y_2 are most probable?*

Let k be a hyperparameter which gives the upper bound on the number of (not necessarily nonzero size) price levels y_i we will compute g_θ^i, on both sides of the order book, i.e. for $i = 1, 2$. If we are only considering changes in the range $-k \leqslant 0 \leqslant k$ for some $k \in \mathbb{N}$, then it would make sense that we have to compute $(2k + 1)^2$ different outputs because that is how many different combinations of y_1, y_2 values there are. Thus we first compute $2k + 1$ values of g_θ^1, and then we condition on Y_1 being equal to each of those, and compute another $2k + 1$ values of g_θ^2 for each of those conditions, giving us $(2k + 1)^2 = 4k^2 + 4k + 1$ possible choices of (y_1, y_2) to be the most probable.

Question 25. *What are the shapes of the inputs and outputs of the neural networks $f_\theta^1, h_\theta^1, f_\theta^2, h_\theta^2$?*

Question 26. *What is the shape of the input $x \in \mathcal{X}$?*

Let s_b be the batch size, s_s be the sequence length, and s_o be the number of rows in the (truncated, constant size) order book, including both bids and asks.

Remark 27. Note that there is no need to stationarize the input data, because the output of the model is going to be the probability of the price changing levels. This is only partially true. We need to stationarize the targets.

Question 28. *Can one forward call compute $g_\theta^1(x, y_1)$ for all $y_1 \in [-k, k] \subseteq \mathbb{Z}$?*

We will do it in 3 forward calls, one for positive y_1 values, one for negative, and one for the classification network h_θ^1. We treat the ask case first, where we are computing g_θ^1, i.e. $i = 1$. We also treat the special case of computing predictions for when $y_1 > 0$, for now. Recall that $i = 1$ is the ask price prediction, and $i = 2$ is the bid. So we have that $x \in M_{s_b \times s_s \times s_o}(\mathbb{R})$, the space of real, 3-dimensional matrices of shape $s_b \times s_s \times s_o$. And our output is a matrix $F^{1,+} \in M_{s_b \times s_s \times k}(\mathbb{R})$, where the superscript is i, the index representing output of either the bid or ask conditional probabilities, just as in the notation for $f_\theta^{1,+}$. Then note that $F_{1,1,j}^{1,+} = f_\theta^{1,+}(x,j)$ for $j > 0$, where $F_{1,1,j}^{1,+}$ represents the j-th element of the first column of the first row of F, where $1 \leqslant j \leqslant k$. Note that the first 1 subscript (indicating the first row) is along the batch dimension, and the second 1 subscript (indicating the first column) is along the sequence/time dimension. The third dimension indicates orderbook level. Hence we have

$$F_{1,1}^{1,+} = \begin{bmatrix} f_\theta^{1,+}(x,1) \\ f_\theta^{1,+}(x,2) \\ \vdots \\ f_\theta^{1,+}(x,k) \end{bmatrix}. \tag{16}$$

Similarly, we have $F^{1,-} \in M_{s_b \times s_s \times k}(\mathbb{R})$ as well, with

$$F_{1,1}^{1,-} = \begin{bmatrix} f_\theta^{1,-}(x,-1) \\ f_\theta^{1,-}(x,-2) \\ \vdots \\ f_\theta^{1,-}(x,-k) \end{bmatrix}. \tag{17}$$

Remark 29. We are currently assuming there are two neural networks for computing nonzero y_1 values given $i = 1$, but we could just use the same network for $f_\theta^{1,+}, f_\theta^{1,-}$. Is this a good idea?

The case where $i = 2$ gets a bit more complicated, since we are conditioning on Y_1. In this case, just computing the distribution over the positive y_2 values for now, our output is $F^{2,+} \in M_{s_b \times s_s \times 2k^2+k}(\mathbb{R})$. Our dimensionality is $2k^2 + k$ since we have $2k+1$ possible y_1 values to condition on, and we have k positive y_2 input values to compute for each of them. Thus, looking at a single sequence element from a single sequence in a batch, we have

$$F_{1,1}^{2,+} = \begin{bmatrix} f_\theta^{2,+}(x,-k,1) \\ \vdots \\ f_\theta^{2,+}(x,-k,k) \\ \vdots \\ \vdots \\ f_\theta^{2,+}(x,k,1) \\ \vdots \\ f_\theta^{2,+}(x,k,k) \end{bmatrix}. \tag{18}$$

But this is gross, since we have what is really two-dimensions collapsed into one. Alternatively, we could implement it such that $F^{2,+}$ has an extra dimension. The raw output of the transformer is passed through a post-decoding layer anyway, which is used to transform the dense hidden dimension data used within the network back to the shape of the raw input, which is in $M_{s_b \times s_s \times s_o}(\mathbb{R})$. Thus the output of the post-decoding layer will have shape $s_b \times s_s \times s_o$, but we could change this. There is no rule that says the output of the post-decoding layer must have the same shape as the input. We can then pass this output through an "F"-layer which reshapes it to be in either $M_{s_b \times s_s \times k}(\mathbb{R})$ or $M_{s_b \times s_s \times 2k+1 \times k}(\mathbb{R})$, depending on whether $i = 1$ or 2. Alternatively, we could just do this all in a single post-decoding layer. Thus we would have

$$F_{1,1}^{2,+} = \begin{bmatrix} f_\theta^{2,+}(x,-k,1) & \dots & f_\theta^{2,+}(x,-k,k) \\ \vdots & & \vdots \\ f_\theta^{2,+}(x,0,1) & \dots & f_\theta^{2,+}(x,0,k) \\ \vdots & & \vdots \\ f_\theta^{2,+}(x,k,1) & \dots & f_\theta^{2,+}(x,k,k) \end{bmatrix}. \tag{19}$$

Remark 30. Remember to add activation functions (`GeLU`) in between all the pre-encoding/post-decoding fully connected layers.

In addition to the F^1, F^2 matrices, we also compute, on each training iteration, the outputs of the h_θ^1 and h_θ^2 networks. Note that h_θ^1 takes as input only x, and outputs a matrix $H^1 \in M_{s_b \times s_s \times 3}(\mathbb{R})$, since h_θ^1 gives a distribution three possible classification labels. Similarly, h_θ^2 takes as input x and y_1, and outputs a matrix $H^1 \in M_{s_b \times s_s \times 2k+1 \times 3}(\mathbb{R})$.

Thus on every training iteration, we must make six forward calls to **six different models**. These compute the matrices $F^{1,+}, F^{1,-}, H^1, F^{2,+}, F^{2,-}, H^2$.

Question 31. *What value do we choose for the hyperparameter k, the upper bound on the number of price levels y_i at which we compute g_θ^i?*

Recall that Y_i is a random variable giving the future best price at time $t + \Delta t$. We wish to compute a confidence interval for this statistic. Well, suppose we knew that the true future best price was sampled from a Gaussian with mean μ and standard deviation σ, then given a confidence level, say 95%, we could compute an interval around the mean μ within which we expect a sample drawn from this distribution to fall exactly 95% of the time. Thus it is sufficient to compute the sample mean and sample standard deviation, and assume these are good estimators of the population parameters.

From visually examining the best ask and best bid delta distributions, we conclude that the data is approximately normally distributed, and thus the assumption above is reasonable.

Question 32. *Should we include the price levels as well as the volume at the levels?*

Yes.

Question 33. *How many levels should we include?*

50 on each side, by heuristic.

Question 34. *Should we normalize the price levels?*

Yes.

Question 35. *How should we normalize the price levels?*

We could subtract mid price from all price levels, or just use integer levels from zero, which represents the best price.

We should use integer levels from zero.

Question 36. *Should we include zero-volume levels?*

No, it is inefficient.

Question 37. *How can we capture shifts along the level-dimension (say a new best ask price pops below the current best ask)?*

We could include integer level labels and a list of integer previous level labels, indicating where that price level was along the level-dimension of the previous order book. If we include all zero-volume levels within our bound along the level dimension (say we take 50 levels on each side), then we need only include the change in level from previous time step to current.

Remark 38. We cannot use the method described previously where we transform the orderbook such that all levels appear dense (remove zero-volume levels by adjusting volume for nonzero-volume levels), since this would not make a distinction between having gaps and having no gaps between nonzero-volume price levels. This would prevent accurate prediction of best price jumps to previously zero-volume levels, which account for roughly 70% of all nontrivial jumps according to the data.

This means we must include the price levels, an answer to Question 32.

Question 39. *What shape should the inputs be?*

Each time step, i.e. each element along the sequence dimension in GPST, should be a matrix of shape $(6, 50)$ where each of the 6 rows represent ask level, previous ask level, ask volume, bid level, previous bid level, and bid volume. This will be flattened into a single 300-dimensional vector.

Remark 40. The level information should be measured with respect to the best price on that side of the orderbook. The quantity should NOT be measured in nonzero-volume levels, it is measured in maximally granular levels.

This way, to compute the previous ask level, we simply subtract the price of that level from the best ask price from the previous time step.