Meta-Regularized Deep Learning for Financial Forecasting

Will Geoghegan
willgeo@stanford.edu

Abstract

Meta-learning has shown great promise for machine learning domains in which problems can be formulated as distributions over sets of tasks. Supervised meta-learners struggle when tasks within a task distribution are not mutually exclusive, meaning the mapping from input to label for a task $\mathcal{T}_i$ can be at least partially inferred from the same mapping for a different task $\mathcal{T}_j$. This issue gives rise to *meta-overfitting*, in which rather than adapting specifically to each individual task, the meta-learner memorizes these mappings and in effect reduces the problem to one of standard supervised learning with a single task. This can be combated with *meta-regularization*, which seeks to regularize the meta-learner without interfering with its learned task-specific adaptations.

We explore various approaches to meta-regularization, comparing and contrasting their effectiveness at tackling the memorization problem compared to non-meta-regularized models in the domain of financial market forecasting. This domain is well-suited to the meta-learning paradigm: different assets have fundamental differences in what affects their behavior that is exogenous to any concrete features we might choose to encode, but at the same time there is significant shared structure between tasks when we define a task $\mathcal{T}_i$ as the regression on an asset i's price one quarter in the future. We can then sample any number of training examples specific to i when we sample i from $\mathcal{T}$.

We show that instead of Gaussian noise on the meta-parameters, a learned L2-based penalty on the meta-parameters can have a meta-regularizing effect by limiting the expressivity of the meta-model without inhibiting in any way the adaptive power of the task-specific parameters.

We show that MAML over a simple feedforward DNN can outperform a similar standard supervised network, and that meta-regularization on top of this MAML implementation can further improve performance. We experiment with different kinds of hand-tuned and learned meta-regularizers, and we see this model's output, when trained on strictly fundamental data, can lead even a very simple trading model to above-market backtested returns on held-out assets on a quarterly time horizon. We probe the meta-optimized inner learning rate behavior of MAML, discuss the underlying strengths and weaknesses of various approaches to meta-regularization, and outline potential extensions of this work for general non-mutually-exclusive meta-learning problems.

[1]Code for this project can be found at `https://github.com/wdg3/regularized-meta-learning`. Unfortunately, the dataset is paid and private so it is unable to be shared.

1 Introduction

Supervised meta-learning has recently made great strides in modeling problems in which we need to quickly adapt to a new task with a limited amount of task-specific data. One pitfall experienced by many meta-learning models is that of non-mutual-exclusivity: when labels for a new task can be inferred based on information learned from prior tasks, it is possible for a meta-learner to learn a model that ignores new task input and simply reduces learning to the standard supervised formulation. Previous approaches to solving (or avoiding) the mutual exclusivity probelm have included arbitrary labeling of tasks at sample time and adding Gaussian noise to the meta-parameters without modifying the task-specific parameters. [8] We will show that adding an L2 loss penalty to the mean squared error in the regression context can have a similar meta-regularizing effect. Our derivation of this result is in Section 4: Theoretical Motivation.

Model-Agnostic Meta Learning (MAML) [2] is a particularly successful framework for meta-learning. It will be our jumping off point for exploring meta-regularization. In the original MAML paper, the mutual-exclusivity problem is sidestepped by defining labels arbitrarily within each meta-batch. We will tackle the problem head on, as it is unavoidable in our chosen domain of financial forecasting. We discuss the dataset building and modeling considerations specific to this domain in Sections 3 and 5.

There has been a large amount of research into using deep neural networks to predict financial markets. These markets are highly stochastic, and any successful meta-learning model needs to be able to deal effectively with both high amounts of uncertainty and high amounts of similarity between tasks. Unlike in standard supervised learning, finanical markets are a relatively untapped domain for meta-learning. Our exploration of the application of meta-learning to the domain and the application of meta-regularization to meta-learning will therefore go hand-in-hand; both are essential facets of this project and both provide interesting insights.

We find very promising results on real held-out stock predictions with our meta-regularized models (see Section 6: Experimental Results), beating the market in a down period with a very simple trading rule. There are nevertheless many possible extensions and improvements that can be made to our model; see section 7: Future Work for these.

2 Related Work

Aside from our baseline supervised DNN, we construct all of our experiments within a MAML framework. [2] This gives us an established starting point for experimenting with different learning models and meta-regularization techniques. However, vanilla MAML implicitly requires that tasks be mutually-exclusive[8], meaning that there can be no model that can solve every task in the distribution simultaneously, in order to force the meta-learner to take up task-specific adaptations. There have been several approaches to creating meta-learners that can train on non-mutually exclusive tasks and solve this so-called memorization problem. We model ours most closely after Yin et. al. (2020) [8]. This approach uses a Gaussian $\mathcal{N}(\theta; \theta_\mu, \theta_\sigma)$ with learned parameters per dimension as noise on the meta-parameters. This noise limits the information that can be stored in the weights of the meta-learner θ, effectively forcing the meta-learner to adapt to each task rather than learning a more generalized model that it then uses in a zero-shot fashion when presented with a new task. We take a similar, but simpler, approach to meta-regularization by learning weight-decay penalties β on the meta-parameters θ that are not applied to the task-specific parameters ϕ. The advantage of this approach is the ease of application and differentiability; we also see similar performance to the stochastic approach in our selected domain. The downside in comparison to Yin et. al., 2020 is that we don't have the same rigorous PAC-Bayes bound on generalization guaranteed by the Gaussian approach, but ours can be informally viewed as a incentive against overt task memorization.

Many approaches have avoided the meta-overfitting issue altogether by using labeling schemes that force tasks to be mutually exclusive. [5][2] Our data simply shares too much latent structure for this approach to be feasible. For example, within the Omniglot context, characters can be assigned arbitrary labels at task sample time. This works because θ can learn extensive information about what makes a character unique, but cannot learn any specific label mappings.

On the domain side, there has of course been much work on the application of deep learning models to financial forecasting problems. [3][6][7] While there have been varying levels of success, what none

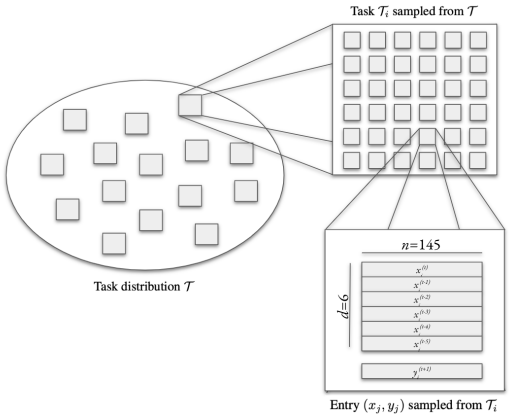

Figure 1: The problem formulation. Tasks corresponding to individual companies are sampled from the distribution of all companies, and time series of fundamental data with corresponding labels are sampled from each task.

of these has yet done is formulate the problem as a meta-learning one. This is the key contribution of this work: the insight that different assets share enough structure and yet differ enough in dynamics to make the meta-learning formulation a natural and effective one for this problem. We have been unable to find any prior work that employs modern meta-learning techniques for the domain of financial forecasting.

3 Data Processing and Problem Formulation

It is universally true in machine learning that a model is only ever as good as the data it's trained on. This is perhaps especially true when it comes to financial markets: the learnable signal is so low relative to the noise that a poor data input will lead to no learning at all. We have chosen to focus solely on fundamental data, i.e., quarterly financial statements for publicly traded companies, for our input, while basing the label on the relative change in price. We similarly use a relatively long time horizon of one fiscal quarter. This allows more time for the highly stochastic day-to-day movements of prices to level out, letting the learnable signal come through.

We source our data from Quandl's Sharadar Core US Equities Bundle.[2] This is a paid dataset that contains daily and quarterly datapoints for all publicly traded US companies from 2009 onward. We begin constructing our dataset by joining all daily price-related datapoints from days when quarterly reports were released for a given company to the appropriate quarterly datapoint. This results in a 145-dimensional vector, with 131 fundamental features such as debt-to-asset ratios, cash flow, cash on hand, etc., and 14 technical features such as price, earnings per share, price/equity ratio, etc. Each input point x is further defined as a time series $\in \mathbb{R}^{6\times 145}$ of 6 quarters worth of data. This allows us to construct the temporal trend and extrapolate it to the future for prediction. All input dimensions are normalized to zero-mean and unit variance in pre-processing, and companies without enough data to do at least 5-shot learning are discarded.

How to define a task for this domain is an open-ended design decision; our decision to define a task as the forescasting of a specific company's price is by no means the only, or necessarily the most effective approach. But it is a straightforward formulation that makes intuitive sense. There is a clear delineation by which elements sampled from within the same task share certain features, like the scale of market capitalization or assets, that those sampled from other tasks don't necessarily share. Other formulations that could make sense include defining a task to be forecasting companies of a specific sector, industry, size, or even country.

Another design decision made in the data pipeline was the time horizon of 1 quarter. It is very possible that looking further into the future would create a higher signal-to-noise ratio, making this meta-learning problem easier from an information theoretic standpoint. But this would reduce

[2] `https://www.quandl.com/databases/SFA/data`

the amount of available data, as well as reduce the potential applicability of the model to making trading decisions. Another decision that struck a balance between total information per datapoint and available data was that of using 6 quarters of backward data instead of 8-12. This significantly increased the amount of eligible companies, because the quarterly data in Quandl's provided dataset only dates to 2013. We ended up with 2352 eligible tasks representing 64368 datapoints. We evenly split companies currently listed in the SP500 into our meta-test and meta-validation sets, and kept the rest as our meta-training set. This ensures a good test for the meta-learning approach: our meta-validation and meta-test sets are explicitly out of sample for our training set, meaning our meta-learner needed to adapt specifically to each new task at test time.Code was processed using the `pandas` and `numpy` Python libraries.

4 Theoretical Motivation

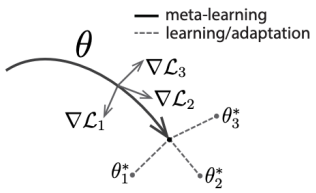

Figure 2: The MAML diagram (Finn et. al., 2017)[2]

In the MAML approach, we represent our model as a function f parametrized by θ. For each task $\mathcal{T}_i$, we sample a batch of K examples. Using these examples, we perform gradient descent on the batch to calculate

$$\phi_i = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f)$$

for the case where we only perform one step, or

$$\phi_{i,j} = \phi_{i,j-1} - \alpha_j \nabla \mathcal{L}_{\mathcal{T}_i}(f)$$

for j gradient descent updates.[2] In our implementation, we learn a separate α for each layer's weight and bias variables, which is optimized via gradient descent in the outer loop. This outer optimization is expressed as, for task distribution $p(\mathcal{T})$:

$$\min_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\phi_i}) = \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f))[2]$$

This is in turn gives us the meta-optimization outer update procedure, with outer learning rate γ, for at timestep t:

$$\theta^{(t)} = \theta^{(t-1)} - \gamma \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\phi_i}).[2]$$

Our objective for this project is to add meta-regularization to this meta-optimization procedure. Intuitively, this means we want to limit the information stored in θ so that no task memorization occurs and task-specific adaptions for task i must instead happen in ϕ_i. Yin et. al. [8] show that we can find an upper bound for this information on the labels contained by the inputs for task $\mathcal{T}_i = (\mathcal{Y}_i, \mathcal{X}_i)$:

$$I(\mathcal{Y}_i; \theta | \mathcal{X}_i) = \mathbb{E}[\log \frac{q(\theta|\mathcal{M})}{q(\theta|\mathcal{X}_i)}] \leq \mathbb{E}[D_{KL}(q(\theta|\mathcal{M})||r(\theta))][8]$$

By modeling the randomness inherent in θ with a Gaussian $q(\theta) = \mathcal{N}(\theta; \theta_\mu, \theta_\sigma)$, they bound the information held in pre-inner-loop θ by adding Gaussian noise with learned parameters per weight parameter.

Our approach to meta-regularization is different from that of Yin et. al. in that rather than stochastically adding noise to θ to limit information flow from θ to ϕ_i, we use an L2-based penalty on the prior weights in θ to empirically limit the expressivity of θ without limiting the ability of the model to aggressively adapt ϕ_i to the task at hand. Concretely, with our mean-squared error based objective, this brings us from the standard MSE loss

$$\mathcal{L}_{\mathcal{T}_i}(f_{\phi_i}) = \frac{1}{K} \sum_{x_i^{(j)}, y_i^{(j)} \sim \mathcal{T}_i} \|f_{\phi_i}(x_i^{(j)}) - y_i^{(j)}\|_2^2[2]$$

to the novel L2 meta-regularized MSE loss

$$\mathcal{L}_{\mathcal{T}_i}(f_{\phi_i}, f_\theta) = \frac{1}{K} \sum_{x_i^{(j)}, y_i^{(j)} \sim \mathcal{T}_i} \|f_{\phi_i}(x_i^{(j)}) - y_i^{(j)}\|_2^2 + \beta\|\theta\|_2,$$

By penalizing the norm of θ but not the norm of the associated post-adaption ϕ_ks, we can achieve a similar meta-regularization effect to that in Yin et. al. Moreover, in plugging this back into the meta-objective, we obtain

$$\min_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\phi_i}) = \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f)) \tag{1}$$

$$= \frac{1}{K} \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \Big(\sum_{x_i^{(j)}, y_i^{(j)} \sim \mathcal{T}_i} \|f_{\phi_i}(x_i^{(j)}) - y_i^{(j)}\|_2^2 + \beta\|\theta\|_2 \Big) \tag{2}$$

$$= \frac{1}{K} \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \Big(\sum_{x_i^{(j)}, y_i^{(j)} \sim \mathcal{T}_i} \|(f(x_i^{(j)}) - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f)) - y_i^{(j)}\|_2^2 + \beta\|\theta\|_2 \Big) \tag{3}$$

$$= \frac{1}{K} \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \Big(\sum_{x_i^{(j)}, y_i^{(j)} \sim \mathcal{T}_i} \|f(x_i^{(j)}) - 2\alpha(f(x_i^{(j)}) - y_i^{(j)}) - y_i^{(j)}\|_2^2 + \beta\|\theta\|_2 \Big) \tag{4}$$

This results in an easily differentiable meta-regularized objective function that we can apply to non-mutually-exclusive task distributions with reduced fear of memorization, without impacting the task specificity of ϕ. Even more enticingly, we can learn our β, or even learn different βs per dimension of our θ. These β variables are optimized by the outer update loop.

The biggest pitfall of our approach is that the L2 component of the gradient of the loss with respect to θ will generally be strongly positive. We must therefore constrain our β variables to be positive. If a β is optimized to be negative (which will almost always happen without this constraint), we can see exploding negative gradients for θ which will in turn lead to exploding weights.

5 Modeling

We ran experiments on several architectures with and without the meta-regularization scheme defined in the previous section.

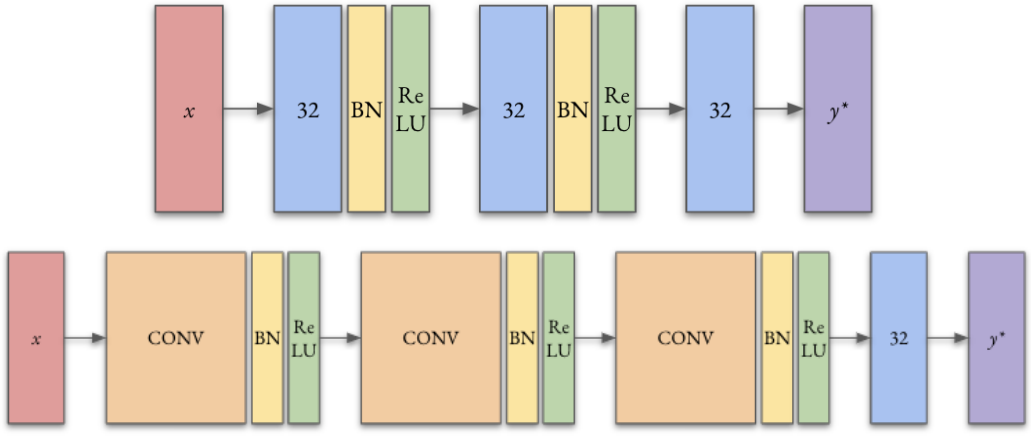

Figure 3: The feedforward and convolutional architectures for testing various meta-regularization schemes. Blue layers represented densely connected layers with 32 hidden units. Orange layers represent temporally convoluted layers with 32 filters. Yellow layers represent batch normalization layers, and green layers represent ReLU activation layers.

Figure 3 shows the basic building blocks of the inner architecture over which MAML is meta-learning in our experiments. All models were written in and trained using TensorFlow.[1] We use CS330's Homework 2 as a basic starting point for our MAML implementation, but the underlying models are specially designed for this problem and we made many adjustments to and experimentations with the meta-learning procedures themselves. The outer loop employs an Adam optimizer [4] with a learning rate of 0.001. We trained the following models on our dataset ($K = 3$, inner loop step for MAML-based models):

- Standard supervised feedforward network (implemented simply by setting the number of inner loop steps to zero)
- Non-meta-regularized MAML over feedforward network
- Non-meta-regularized MAML over convolutional network
- Meta-regularized MAML over convolutional network ($\beta = 0.1$)
- Meta-regularized MAML over convolutional network (single learned β)
- Meta-regularized MAML over convolutional network (learned βs per parameter)
- Meta-regularized MAML over feedforward network (learned βs per parameter)

We chose to run the above experiments because they give a broad picture of how well this meta-regularization scheme works, how effective meta-learning our β parameters is, and the simplicity of the inner architecture ablates it as a question mark in interpreting our results. The results of the above experiments are described and investigated in the next section.

6 Experimental Results

All models were trained for 10000 outer loop iterations on an NVIDIA T4 GPU via Google Cloud Platform. Wall training time per run took anywhere from 90 minutes for the standard supervised model to 7 hours for the MAML convolutional models. A meta batch size of 32 was used for all runs. Inner learning rate variables and learned β regularization constants were all initialized to 0.1.

There are several interesting observations to be made about the MSE training curves in Figure 4. Most obviously, we see that the convolutional models converge (or come closer to convergence) far more rapidly than the feedforward models. Additionally, meta-regularization makes a much more significant difference in meta-validation MSE for the feedforward models than for the temporally convoluted models. When taken together, these facts seem to imply that the feedforward models struggle with the difficult second-order optimization problem present by MAML. We see that meta-regularization (pink curve) makes an especially large difference on the feedforward model in speed of convergence compared to the base version (purple curve).

One interesting detail is that the supervised feedforward architecture (no MAML) has the lowest MSE, but also the lowest two-way accuracy. While initially puzzling, this just indicates that this version is much better tuned numerically to handle outliers, which drive up the MSE on the MAML architectures. This is another confirmation of the difficulty of the second-order optimization.

We see again on the two-way accuracy curves that meta-regularization helps the feedforward MAML model significantly, but now this effect is also clear for the convolutional model. All meta-regularized models seem to cluster quite tightly around the 70% mark. As this is a very noisy problem, it is quite possible that this is approaching Bayes optimality given the data we have (which is nowhere near perfect information).

TensorFlow/Tensorboard's histogram and distribution writer functionality allows us to dig a little deeper into what is happening at the meta-optimization level. In Figures 5 and 6, we can see how the distribution of inner learning rate variables was optimized over time. Again we see the theme of convolutional models converging far more quickly. The most interesting detail here is that some αs for the feedforward meta-regularized model were temporarily negative.

It is difficult to reason about the implications of a negative inner learning rate. One possibility is that it simply overshot 0 due to a too-large outer learning rate. But this doesn't quite seem to be the case, as these variables entered and left negative territory over the course of thousands of iterations. Intuitively it seems as if this could have a regularizing (though not *meta*-regularizing) effect by walking back aspects of ϕ that have overfit to the training task, but this requires deeper investigation.

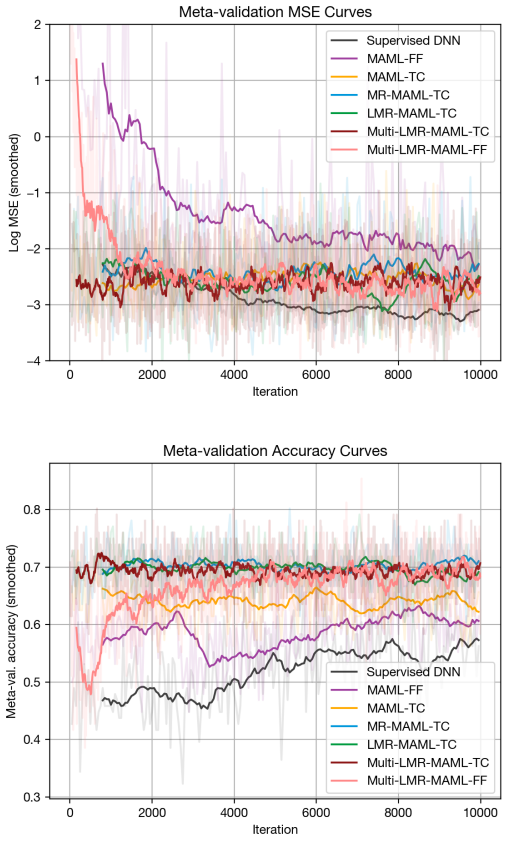

Figure 4: Error and accuracy curves on meta-validation set. While this is a regression problem, we are interested in the 2-way accuracy for this specific domain.

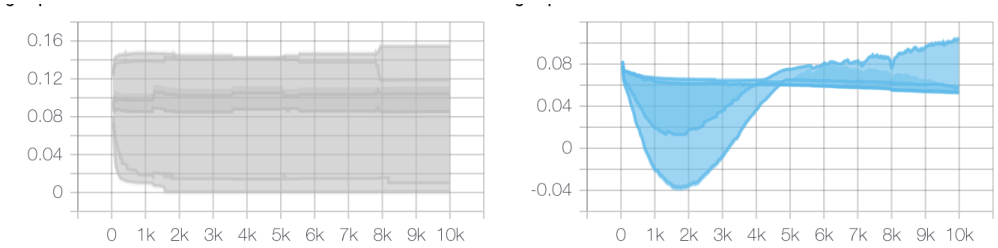

Figure 5: Distribution of α (inner learning rate) variables over time for meta-regularized convolution model (left) and meta-regularized feedforward model (right).

In the results in Figure 7, we see all meta-regularized models outperform their non-meta-regularized counterparts. **All meta-regularized models also beat the market for the given quarter with the simple returns strategy.** It is worth noting the comparably worse performance of the weighted metric. This is because the MAML models are poorly tuned for handling outliers; this leads to worse performance when outliers are emphasized. There is relatively little difference between the meta-regularized models, but perhaps that we didn't train the model with multiple learned β variables long enough to overcome this added complexity. We also see that the convolutional models consistently outperform the feedforward counterparts.

An interesting minor point is that the convolutional models all predicted more than 90% of the test samples to fall in price from the end of Q2 2020 to the end of Q3 2020. It is reasonable to conclude that the model has learned how to approximate macroeconomic conditions from indicators such as

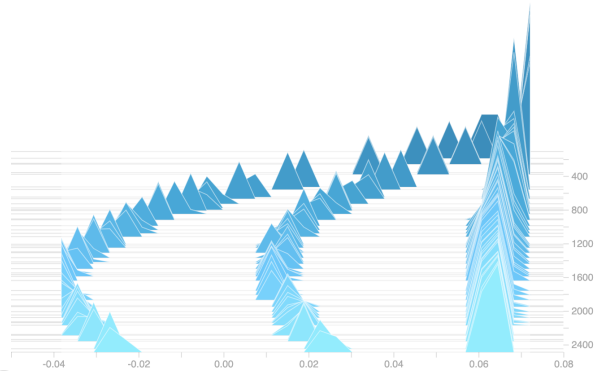

Figure 6: Another visualization of how the distribution of learning rates changes through training iterations.

Model	Meta-test MSE	2-way Meta-test Acc.	Simple Return	Weighted Return
Supervised DNN	0.032 (0.04)	59.83% (0.26)	-5.93%	-5.50%
MAML-FF	0.012 (0.01)	63.03% (0.27)	-1.16%	-2.88%
MAML-TC	0.016 (0.04)	71.37% (0.22)	-0.65%	-3.32%
MR-MAML-TC	0.012 (0.02)	71.57% (0.22)	**-0.25%**	-2.16%
LMR-MAML-TC	**0.011 (0.01)**	**72.37% (0.22)**	-0.30%	**-1.88%**
Multi-LMR-MAML-TC	0.012 (0.02)	69.88% (0.22)	-0.30%	-1.98%
Multi-LMR-MAML-FF	0.014 (0.03)	64.96% (0.27)	-0.97%	-2.27%
Actual	N/A	N/A	-6.25%	-6.25%

Figure 7: Q2-Q3 2020 meta-test return on SP500 companies. Simple return assumes even allocation, weighted return weights by regression prediction value. **NB: A bug in return calculations produced different results in the table used during project presentation.**

cash flow, which is included in our fundamental data, as businesses were indeed cash-strapped to a historical degree in the middle of 2020.

Overall, the results are promising. Outperforming the market is always a solid benchmark, especially with a trading rule as simple as the one outlined above, and we outperformed the market in the test quarter significantly. It is unfortunate that the test quarter was a down one across the market, but we avoid calculating returns on earlier quarters because the market-wide correlations could cause information leakage from meta-train to meta-test set, and we need to avoid that.

7 Future Work

As previously stated, there are a multitude of potential extensions of and/or improvements to this project. First and foremost, our development of a theoretical basis for using a meta-L2 penalty for regularization is very rough. Further theoretical and empirical investigation of this approach could shed further light on how it compares to the stochastic approach of Yin et. al. [8] This is the most interesting avenue that this project has been opened up, and I would be very interested in exploring it more in the future.

On the implementation side, the most obvious shortcoming of this project is extreme simplicity of the inner network architecture for all of our experiments. It made sense from an available compute and time perspective, but it is hard to say whether we approached Bayes optimal performance for our data, or our models simply all lacked enough expressive power to break through the 70% 2-way accuracy barrier. More sophisticated models combined with more complete and robust data would hopefully be able to produce a meta-learner that could generate consistently higher above-market returns.

Speaking of our data, there are always improvements to be made to the data pipeline. We use a simple dataset construction, and someone with more specific financial markets domain knowledge would be well-equipped to greatly improvement on our setup.

Finally, I am greatly intrigued by the phenomenon of negative learning rates in the inner loop of MAML. Unlike the issue of negative L2 penalties, these do not lead to numerical instability. Rather, it even seems possible that they can have a regularizing effect, serving as a backpedal for the task-specific learner when the meta-learner has overindexed into a certain parameter. This is another avenue that would be very interesting to explore.

8 Conclusion

In this project, we have tackled the problem of non-mutual-exclusivity in supervised meta-learning. We have explored a novel approach to meta-regularization to solve this problem, provided a theoretical motivation for it, and applied it to the domain of financial forecasting with positive results. Our meta-regularized models outperformed both their non-meta and non-meta-regularized counterparts, as well as the market itself for the backtesting period.

While we have shown promising results for this project, it has by no means exhausted all potential avenues of exploration. We have outlined a few potential extensions in the Future Work section, and look forward to exploring them ourselves.

There is also great potential for different formulations of financial market forecasting as meta-learning. Rather than individual companies as tasks, companies can be further grouped into industries or sectors, broken down further temporally, etc.

The meta-overfitting problem is a very general one within the field of supervised meta-learning; there are many other applications of meta-regularization outside of financial markets. We have found one valid technique for tackling it, but there is no telling how many more may yet be developed. We hope that we have built on the extraordinary work of others, especially Chelsea Finn and Mingzhang Yin, whose two papers "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks" [2] and "Meta-Learning without Memorization" [8] we cited frequently and which formed much of the basis for this work.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *CoRR*, abs/1703.03400, 2017.

[3] Thomas Fischer and Christopher Krauss. Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research*, 270(2):654–669, 2018.

[4] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[5] S. Ravi and H. Larochelle. Optimization as a model for few-shot learning. In *ICLR*, 2017.

[6] Justin Sirignano and Rama Cont. Universal features of price formation in financial markets: perspectives from deep learning. *Quantitative Finance*, 19(9):1449–1459, 2019.

[7] Avraam Tsantekidis, Nikolaos Passalis, Anastasios Tefas, Juho Kanniainen, Moncef Gabbouj, and Alexandros Iosifidis. Using deep learning to detect price change indications in financial markets. In *2017 25th European Signal Processing Conference (EUSIPCO)*, pages 2511–2515. IEEE, 2017.

[8] Mingzhang Yin, George Tucker, Mingyuan Zhou, Sergey Levine, and Chelsea Finn. Meta-learning without memorization. *CoRR*, abs/1912.03820, 2019.