HIERACHICAL ACTIVE LEARNING BIOINFORMATICS APPLICATION

by

James D. Duin

A THESIS

Presented to the Faculty of

The Graduate College at the University of Nebraska

In Partial Fulfilment of Requirements

For the Degree of Master of Science

Major: Computer Science

Under the Supervision of Professor Stephen Scott

Lincoln, Nebraska

February, 2017

HIERACHICAL ACTIVE LEARNING BIOINFORMATICS APPLICATION

James D. Duin, M.S.

University of Nebraska, 2017

Adviser: Stephen Scott

I made the classifiers to predict the BioDataset, I made the plots for the Active Passive curves, I made the plots for the Fixed Fine Ratio experiments with various costs.

DEDICATION

This thesis is dedicated to my parents Paul and Vicki Duin and fiancee Anna Spady.

ACKNOWLEDGMENTS

I would like to thank my advisor Dr. Stephen Scott for guidance in selecting this research topic, Yugi Mo for his work in developing the HAL methodology, and Dr. Douglas Downey for his work on this topic. I would like to thank Juan Cui, Jiang Shu, Kevin Chiang for assistance accessing and understanding the protein dataset that is the subject of the paper.

Contents

Chapter 1

Introduction

1.1 Machine Learning

Machine Learning algorithms are defined as computer programs that learn from experience E with respect to some class of tasks T and performance measure P, if their performance at tasks in T, as measured by P, improves with experience E [4]. In the context of this paper, the machine learning algorithm that is used is a support vector machine (SVM) implementation by libSvm [3]. The performance measure is the classification of protein instances according to a label, for example, originated in the mitochondria or not. The experience is the number of training instances in the training set. The dataset is initially partitioned into training and test sets. The algorithm improves its classifier structure based on the features of each instance by iterating through all instances in the training set. When training has been completed, the classifier is then tested on the test set and the number of instances that the classifier correctly or incorrectly labels determines its accuracy, precision and recall scores. In our protein dataset there are 20,098 proteins with 449 features each relating to their structure. In our experiments an SVM classifier is applied to the dataset with the goal of achieving high precision scores for the label mitochondrion, that is, if the protein originates in the mitochondria or not.

1.2 Hierarchical Bioinformatics Data Set

The protein dataset is labeled according to where it originates in the cell. At the root is mitochondrion, then there is the sub level labels for if its native to the mitochondria or if it has a separate target compartment specification. The complete tree along with the number of instances belonging to the each label is included

in *Figure* 1.2.

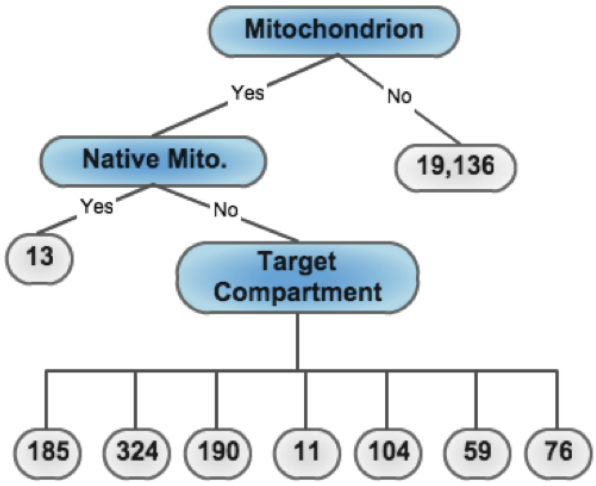

Figure 1.1: The protein dataset is labeled according to where it originates in the cell. At the root is mitochondrion, then there is the sub level labels for if its native to the mitochondria or if it has a separate target compartment specification. The complete tree along with the number of instances belonging to the each label is included in 1.2.

1.3 Coarse Grained vs Fine Grained Trade Off

The classifier that does not take advantage of any of the fine grained labels works off of the root labels for each instance and does not train separate classifiers for the fine grained labels. This classifier is referred to as the coarse grained classifier. The classifier that does use fine grained labels, and trains a separate classifier for each label, then combines them to generate a root level label is referred to as the fine grained classifier. It can be demonstrated through a dummy example that for certain datasets, a fine grained approach to the root level classifier can achieve higher levels of precision for the same level of recall. Such a dataset is shown in 1.3. The classifiers for this dataset can be thought of as a function of axis parallel rectangular boxes. For the course grained to have high recall and return all of the positive circle instances, it must encompass the entire dataset and incidentally return all of the negative diamond instances as positive also. A fine grained approach is preferable for the dummy dataset pictured. It is the intention of this study to demonstrate that the fine grained classification approach for a root level classifier will achieve higher levels of precision for the same level of recall when applied to the protein dataset.

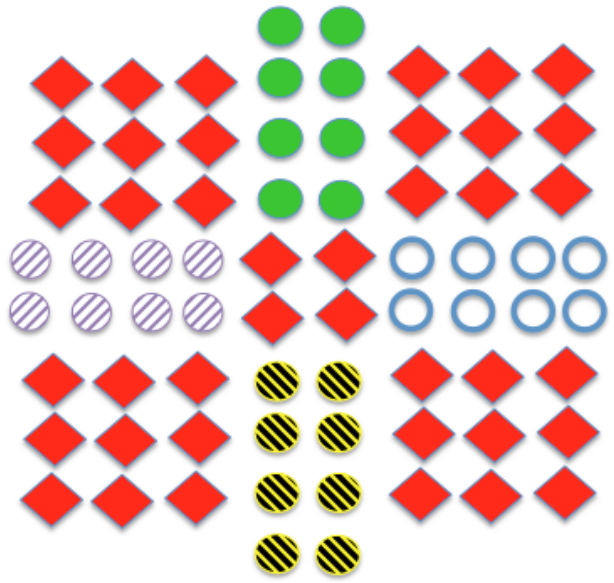

Figure 1.2: Demonstration of a dataset that would benefit from multiple fine grained learners for each circle type. In order for the coarse grain learner to have high recall, precision must be scarified and a large amount of false positives returned. By combining fine grained classifiers the same level of recall can be achieved with a higher level of precision because none of the false positive diamonds will be returned

Chapter 2

Background and Related Work

2.1 Active Learning

Active Learning relates to the coarse grained vs fine grained tradeoff because it is reasonable to assume that fine grained labels may not be as readily available as coarse grained labels, and thus have a higher cost. An active learning approach is used to determine how many fine grained labels to purchase in order to minimize the total cost to train the algorithm an maximize the precision and recall scores. The following equations for precision, recall, and a weighted F score are shown below in equations 1 through 3.
Precisioneqn
Recalleqn
$F0_5 eqn$

The goal in an active learning approach is to maximize the F measure where equals 0.5 [2]. The F-0.5 measure gives more weight to precision, as opposed to recall, so it gives incentive to purchase enough fine grained labels to increase the F-0.5 measure. The coarse grained labels will cost less than fine grained labels, but the increase in the F-0.5 measure justifies the increase in cost up to a certain point. The F-0.5 measure is used in the results section of this paper. ... That's why we use PRauc in the results.

2.2 Other Papers cited by Yugi

Describe some of the other papers that Yugi cited.

2.3 Hierarchical Active Learning

The Hierarchical Active Learning algorithm (HAL) is shown diagrammatically in Figure 3. Multiple fine grained classifiers are trained at each level of the Hierarchy of the dataset. Queries to the oracle are performed purchase the most cost effective labels to add to the training sets of the classifiers. The active learning cycle continues until a cost budget has been reached. The benefit of an active learning approach is to maximize the F-0.5 measure for a given cost budget. It was the goal of this study to apply the HAL algorithm to the protein dataset, however this is not achieved at this time. An existing application of HAL is briefly discussed in the following section.

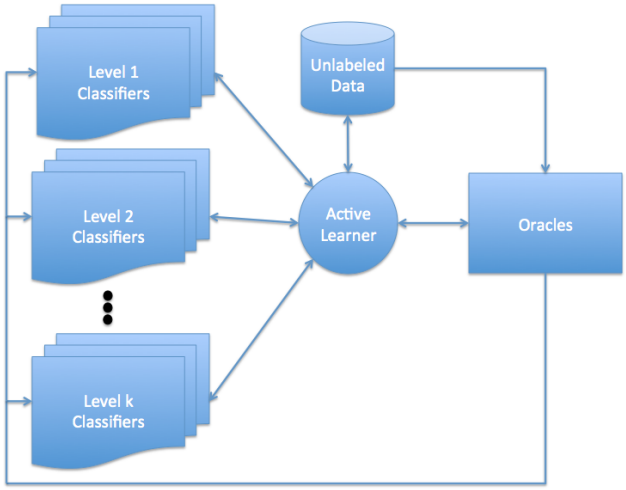

Figure 2.1: Diagram of HAL approach

2.4 Application to Dispatch Dataset

HAL was applied to a Dispatch dataset. This dataset contains 375,026 manually labeled hierarchical names across 1,384 newspaper articles [2]. This is a clear example where fine grained labels have a higher cost since it is easier for a person to manually determine if the article pertains to an organization or not, rather than if it pertains to a railroad or a zoo, which would be sub labels of the organization root. The first analysis step was to determine that the F-0.5 measure is increased by using fine grained classifiers. The results are shown in Figure 4. The highest F-0.5 measure for a given iteration of purchases of training instances is obtained by using the active learning approach with all fine-grained labels. The passive learning curves were generated by selecting batches of instances randomly rather than querying the oracle for a specific

label type that offers that most gain in classifier accuracy. The active learning curves did take advantage of querying for specific labels in order to maximize gain in classifier accuracy. ... add other data sets.

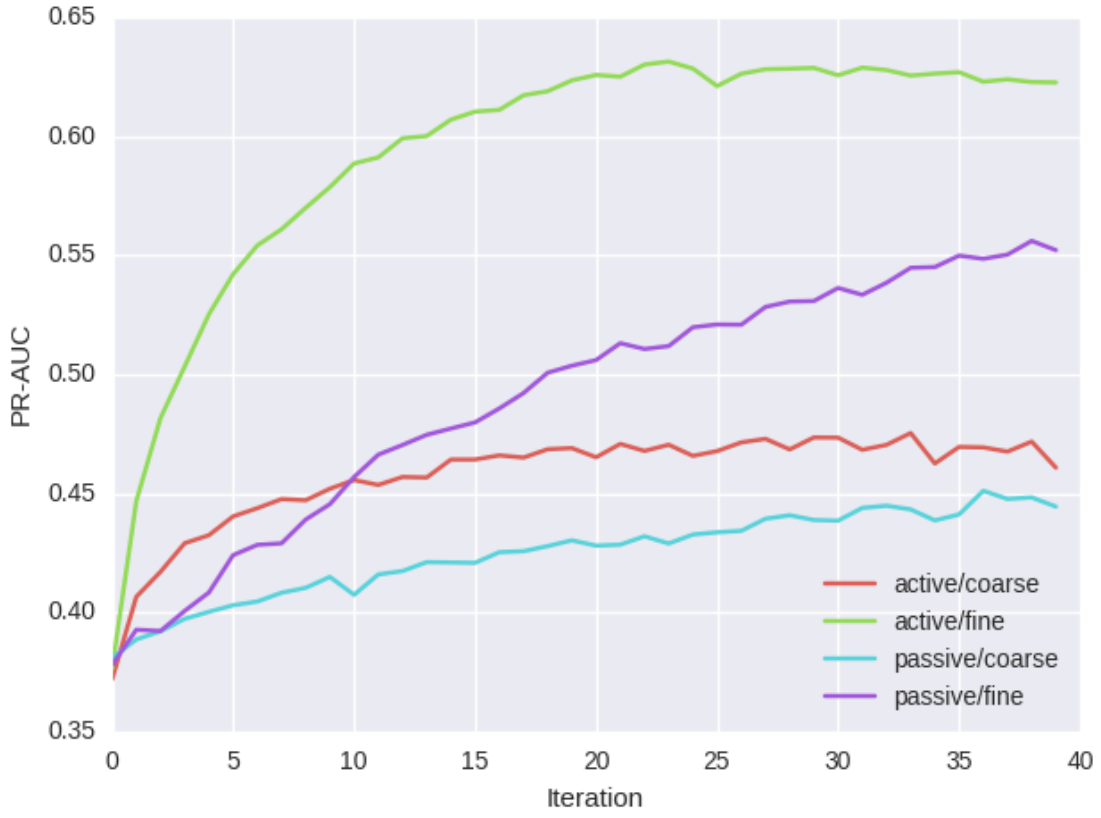

Figure 2.2: Application of HAL demonstrating the benefit of Actively selecting the type of labels to purchase for instances rather than randomly selecting labels to purchase, as in the Passive curves.

The next analysis step is to apply a given ratio of fine grained vs coarse grained labels to purchase at each batch request to the oracle. The results of varying the percentage of fine grained labels purchased are shown in Figure 5. The figure shows that even a small amount of fine grained labels purchased, that is, 20perc provides a significant increase in the F-0.5 measure for a given iteration. ... pr auc results. Add explanation of Bandit approach and results.

Chapter 3

Bio HAL Application

3.1 Training and Testing Coarse Grain and Fine Grain Classifiers

The first step is to examine the dataset.

Classes	All
0	19136
1	13
2	185
3	324
4	190
5	11
6	104
7	59
8	76
Total	20098
Shape	450

Table 3.1: This is what the dataset looks like there are 20098 instances total with 450 features each.

Next the dataset is partitioned into 10 folds, each fold contains a representative proportion of each of the classes, the instances are added to each partition at random. The total partitioning looks like Table 3.2

All	Folds	0	1	2	3	4	5	6	7	8
1	2010	1914	1	19	32	19	1	11	6	7
2	2010	1914	1	19	32	19	1	11	6	7
3	2010	1914	1	19	32	19	1	11	5	8
4	2010	1914	1	19	32	19	1	10	6	8
5	2010	1914	1	18	33	19	1	10	6	8
6	2010	1914	1	18	33	19	1	10	6	8
7	2010	1913	2	18	33	19	1	10	6	8
8	2010	1913	2	18	33	19	1	10	6	8
9	2009	1913	2	18	32	19	2	10	6	7
10	2009	1913	1	19	32	19	1	11	6	7
Total	20098	19136	13	185	324	190	11	104	59	76

Table 3.2: This is what the folds of the dataset look like.

Then 9 of the folds are compressed into the test set and the fold held out is the test set the totals of each class in the train and test set for fold 1 is shown in Table 3.3.

train Total	0 18088	1 17222	2 12	3 166	4 292	5 171	6 10	7 93	8 53	69
test Total	0 2010	1 1914	2 1	3 19	4 32	5 19	6 1	7 11	8 6	7

Table 3.3: This is what the train and test set look like.

How did I run coarse and fine classifiers.

coarse just used marked all positives as 1 and ran a binary classifer.

fine grained classifiers trained 8 separate classifiers for the 8 fine grained classes, so for fine grain classifier for 1, all other fine grained classes are marked as 0 in addition to all the coarse instances being marked as 0.

Also the Pr auc and Roc auc are the primary metrics for determining the performance of the classifier.

The next step is to determine what classifier can be applied to 'learn' the classes of this dataset.

because the experiment will involve running multiple rounds with increasing the instances to be trained on iteratively I tested each classifier against the full dataset and then a reduced dataset with one fifth of the negative instances.

I tried using SVM. Throughout this project I used the python library sci-kit learn [1]. The support vector machine implemented by this library has the following default parameters. SVC C=1.0, kernel='rbf', degree=3, gamma='auto', coefo=0.0, shrinking=True, probability=False, tol=0.001, cache-size=200, class-weight=None, verbose=False, max-iter=-1, decision-function-shape=None, random-state=None.

I tried different scaling methods (min max scaler, std scaler), I settled on std scaler.

I tried different feature select measures, I settled on 75 perc feature select.

I tried different different C costs, kernels, decision function shape, gamma, tolerance settled on classif = svm.SVC(C=1.0, kernel='rbf',decisionfunctionshape='ovo',gamma=0.0025, tol=0.00001)

I left the class weight as balanced for this part, the results did not show much advantage for using the fine grained classifier.

next I tried using a Logistic regression classifier.

I tried different scaling methods, min max scaler, std scaler, I settled on min max scaler.

I tried different feature select measures, decided to use all of the features.

I tried different C costs, tolerances, and class weights. I settled on C=0.1, tol = 0.00001, and weight equal to the balanced, adjusted via a scaling line.

I also further tuned the fine grained classifiers starting from the initial scaling from the line an then multiplying that by a ratio. I got this vector of ratios [0.87, 0.4, 0.78, 0.65, 3.48, 0.78, 1.74, 0.87]

This shows the advantage to fine grained labels to justify the experiment.

coarse-pr	fine-pr	coarse-roc	fine-roc	coarse-acc	fine-acc	coarse-f1	fine-f1
0.898	0.901	0.905	0.896	0.767	0.945	0.259	0.474
0.870	0.869	0.847	0.846	0.803	0.944	0.272	0.456
0.897	0.907	0.895	0.901	0.792	0.947	0.287	0.500
0.864	0.866	0.852	0.848	0.778	0.943	0.256	0.430
0.855	0.865	0.859	0.859	0.795	0.947	0.269	0.451
0.867	0.869	0.874	0.865	0.785	0.939	0.263	0.417
0.871	0.887	0.873	0.881	0.784	0.940	0.269	0.442
0.835	0.845	0.843	0.842	0.794	0.940	0.258	0.388
0.870	0.878	0.869	0.871	0.784	0.939	0.262	0.417
0.873	0.873	0.891	0.890	0.786	0.933	0.279	0.368
avg 0.870	avg 0.876	avg 0.871	avg 0.870	avg 0.787	avg 0.942	avg 0.268	avg 0.434

Table 3.4: Here are the results for the logistic regression passive 10 folds.

3.2 Passive SVM Rbf kernel vs Logistic Reg

add text'example cite'[2]

3.3 Active vs Passive curves

The following plots were obtained with a round batch size of 100 and a starter set of 1040 instances out of the total 2098 instances. The plots are the average of 10 folds, for each fold a test set of 2010 containing representatives of each class was held out, out of the remaining 18088, the starter set was selected which again contained representatives of each class. Coarse and fine classifiers share the same starter set. During each round coarse and fine classifiers are trained on their corresponding sets, metrics are outputted on the held out test set, then confidence estimates are ran on the remaining eligible instances. Eligible instances are kept in separate sets for coarse and fine, 100 of the most uncertain instances are removed from each eligible set and added to its corresponding coarse or fine set to be trained on for the next round.

3.3.1 Plots for Logistic Regression Active vs Passive curves

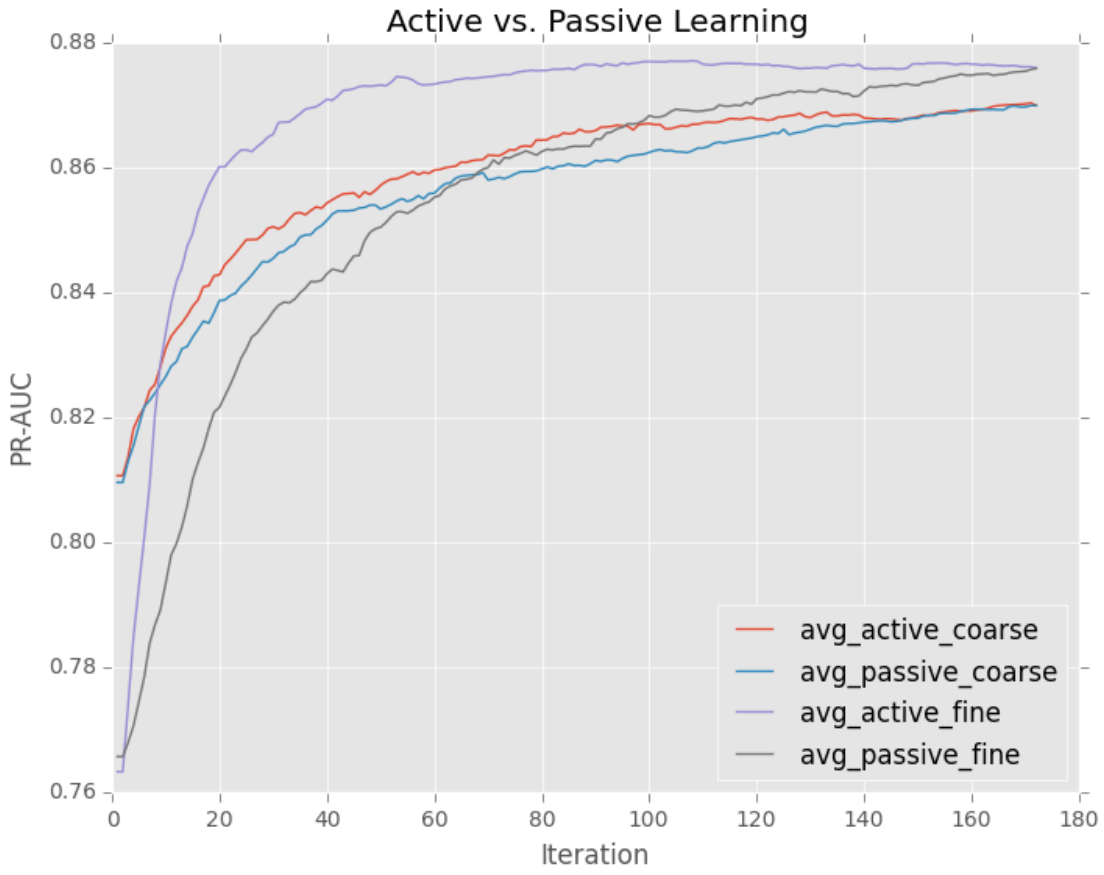

Figure 3.1: The PR AUC curves for rounds with the Logistic Regression classifier conforms to expectations, with active-fine having the highest performance. Active-coarse outperforms passive-coarse. Passive-fine doesn't outperform the coarse classifiers until rnd 100.

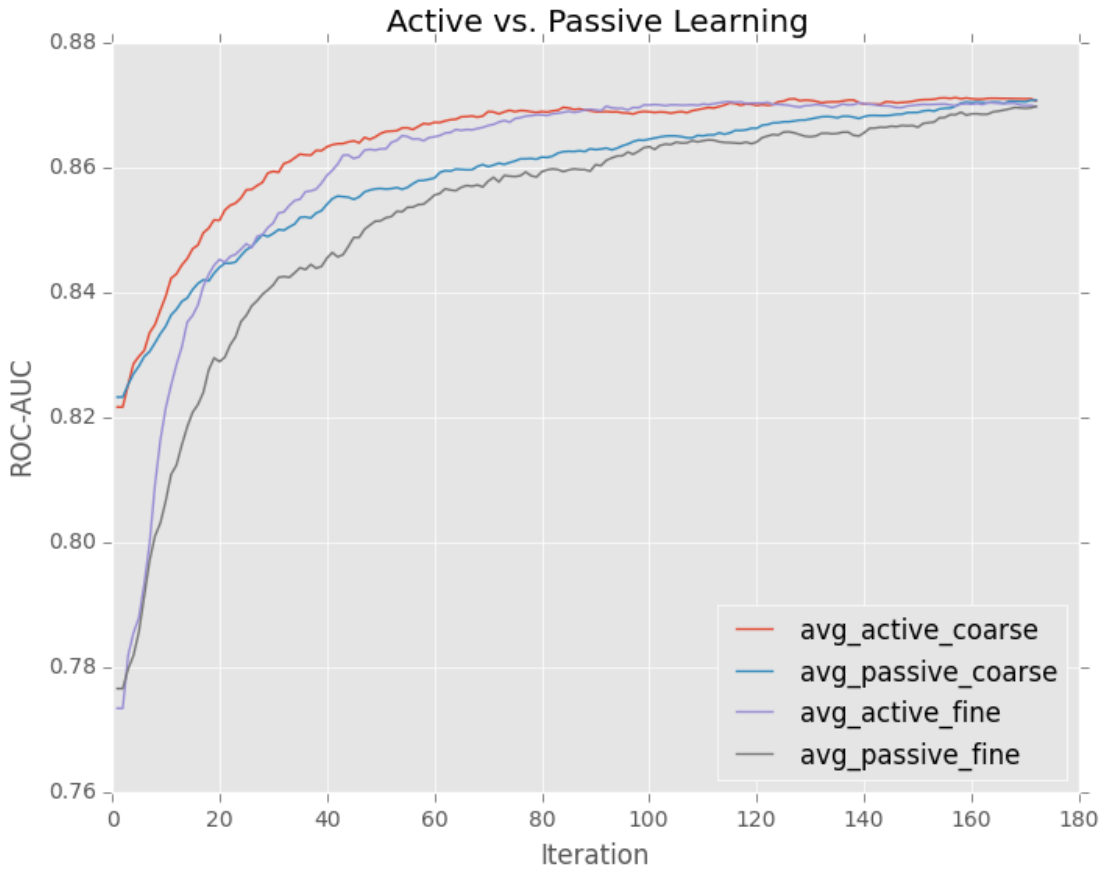

Figure 3.2: The ROC AUC curves for rounds with the Logistic Regression classifier. The active curves beat out the passive curves for both coarse and fine. Coarse roc starts with an advantage over fine as in the PR curves. Both converge to the same rate after roc auc level after 80.

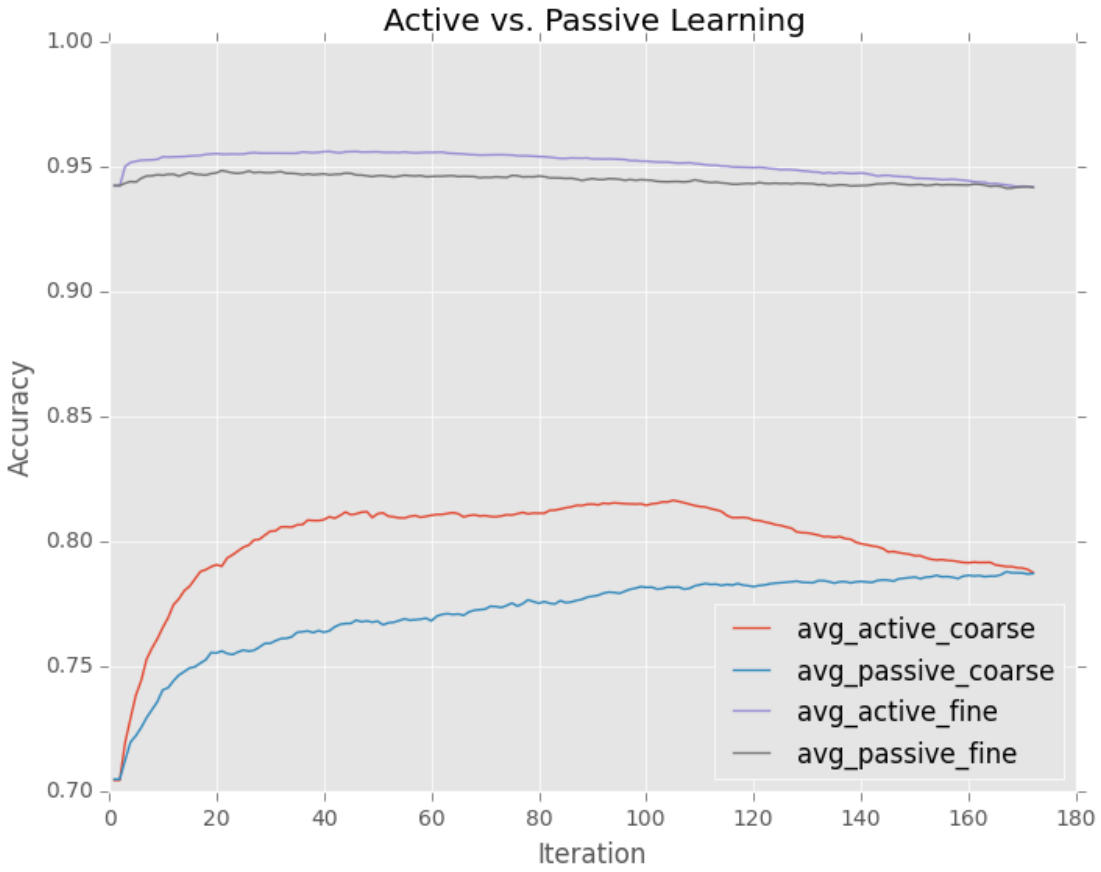

Figure 3.3: The accuracy of the fine classifiers stays at roughly the same rate throughout the rounds, this is due to an effective weighting scheme for the fine grained classifiers. The active coarse accuracy drops towards the end due to an increase in false positives as more negative instances are added in the later rounds.

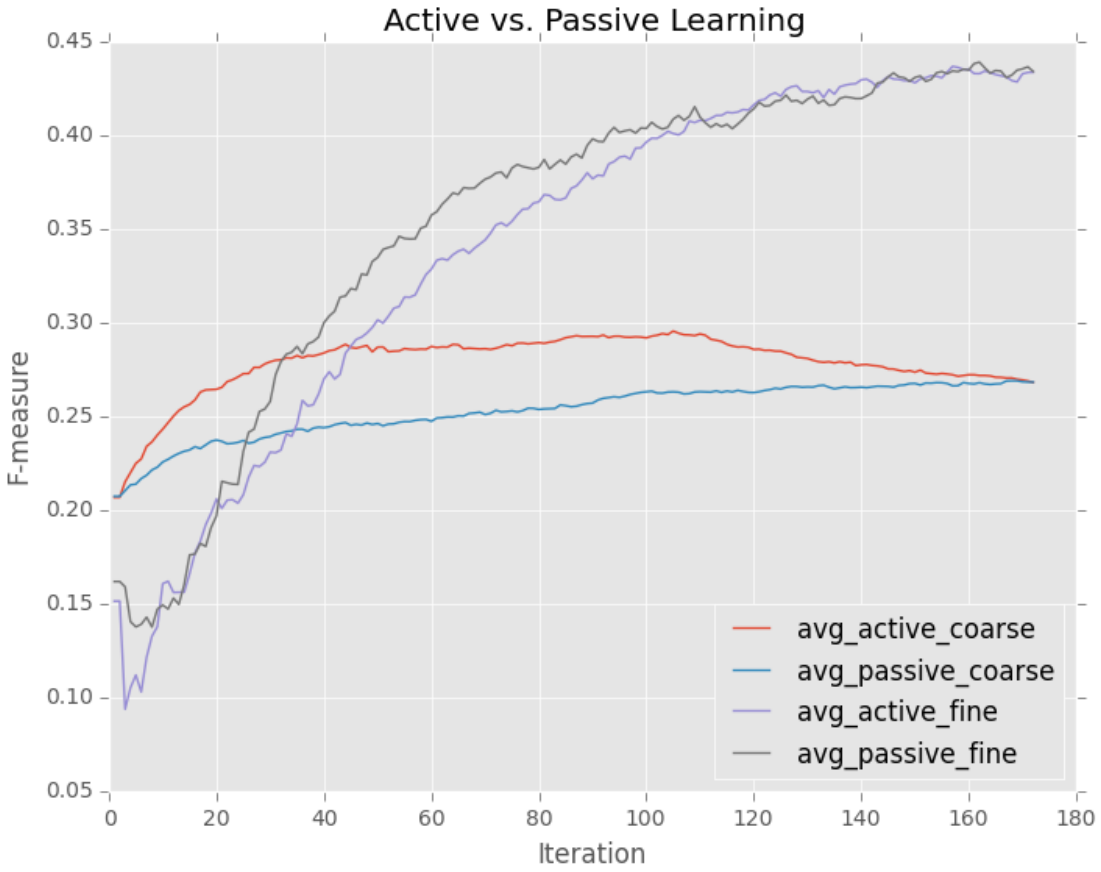

Figure 3.4: The F-measure of the the fine classifiers increases throughout the rounds as more true positives are predicted. The active coarse again decreases at later rounds due to increased false positives.

3.3.2 Plots for SVM Active vs Passive curves

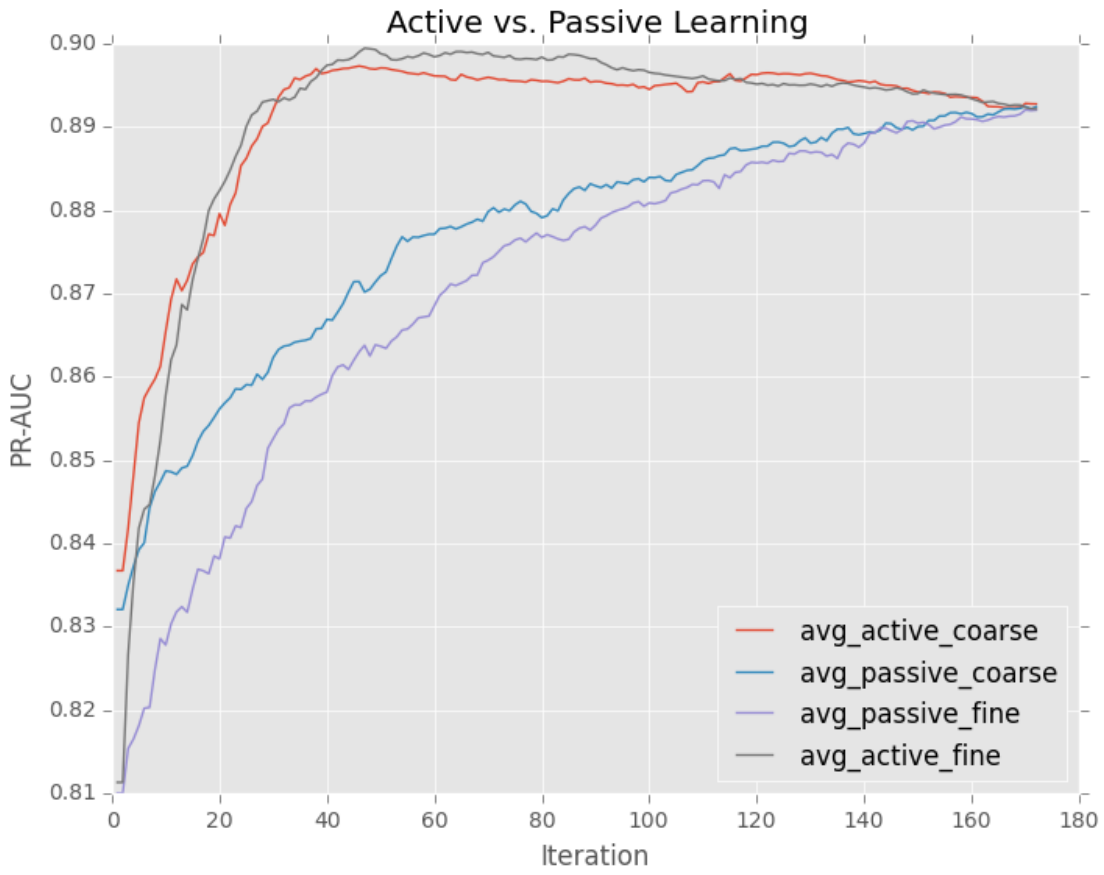

Figure 3.5: The PR AUC curves for rounds with SVM show little advantage for fine. The results are slightly different than the ones shown on 2/14 due to fixing a bug with the code that wasn't performing the preprocessing scaling for the SVM case at the same stage as it was being done for the logistic regression classifier.

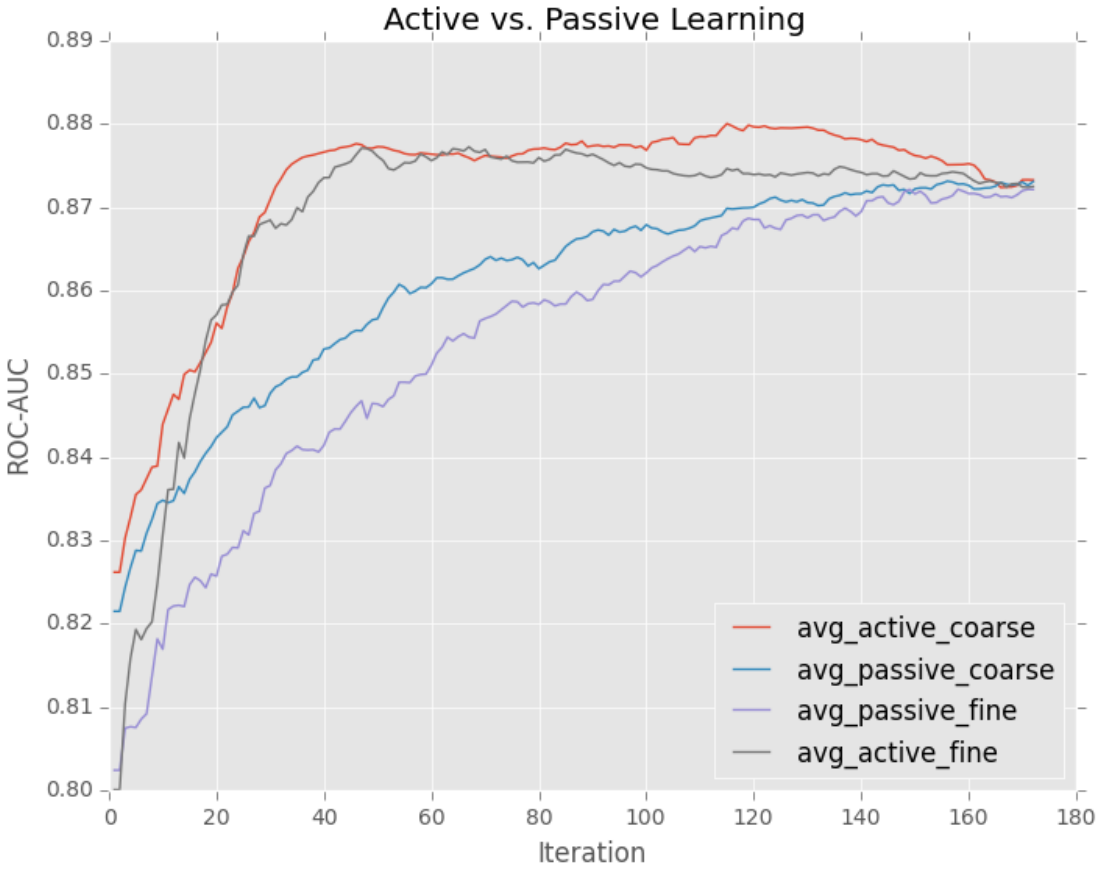

Figure 3.6: The ROC curves show more of an advantage for coarse classifiers.

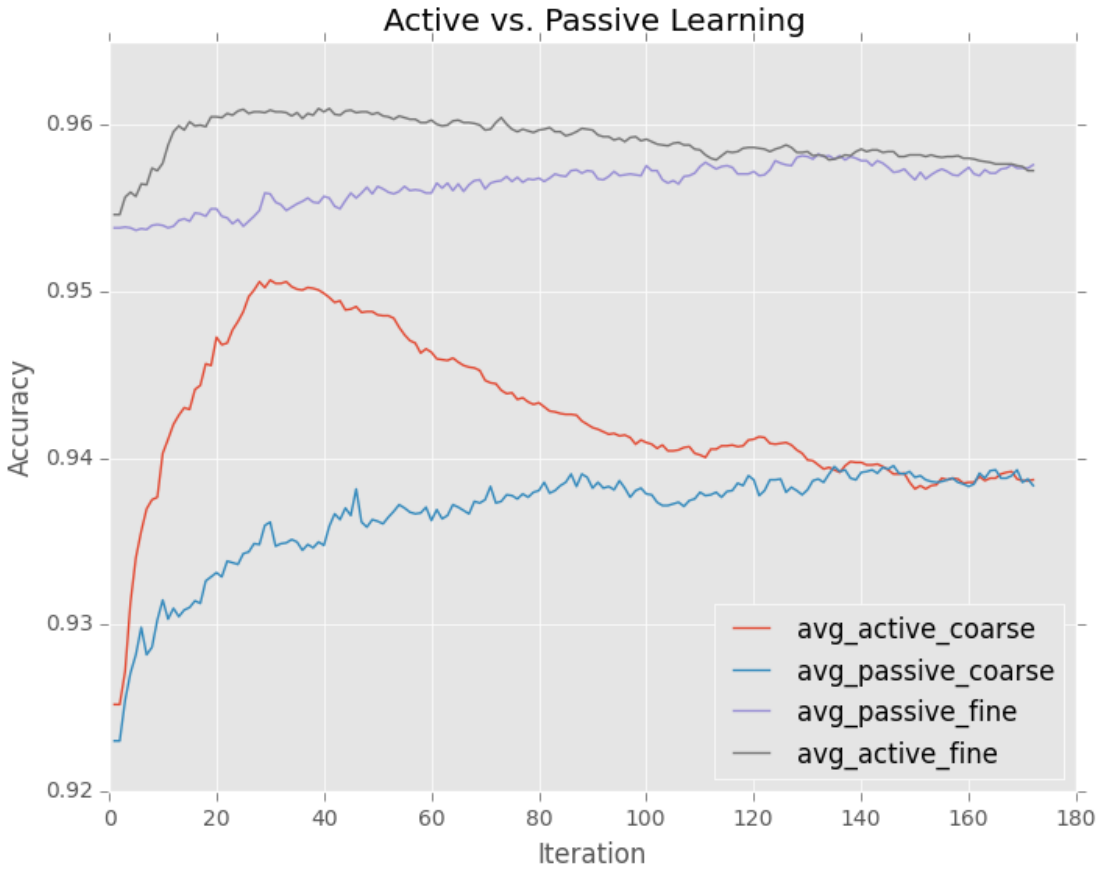

Figure 3.7: The accuracy for the coarse decreases sharply due to coarse predicting steadily more false positives, behaving similar to the Log Reg case. Fine accuracy is higher due to predicting less false positives than coarse. Fine also predicts less true positives, compare apx. 37 to apx. 60 t.p. for coarse at round 60.

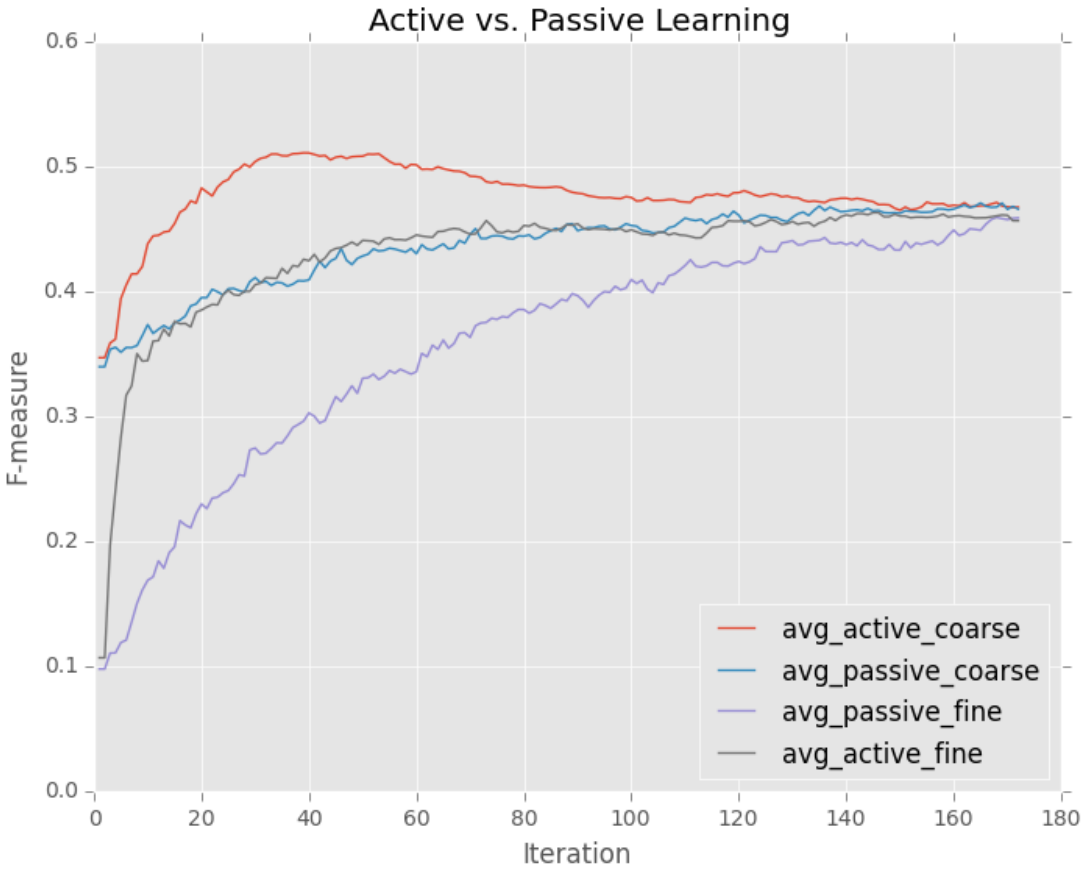

Figure 3.8: The F-measure favors coarse, and trends to the same level for both coarse and fine.

3.4 Plots for FFR experiments

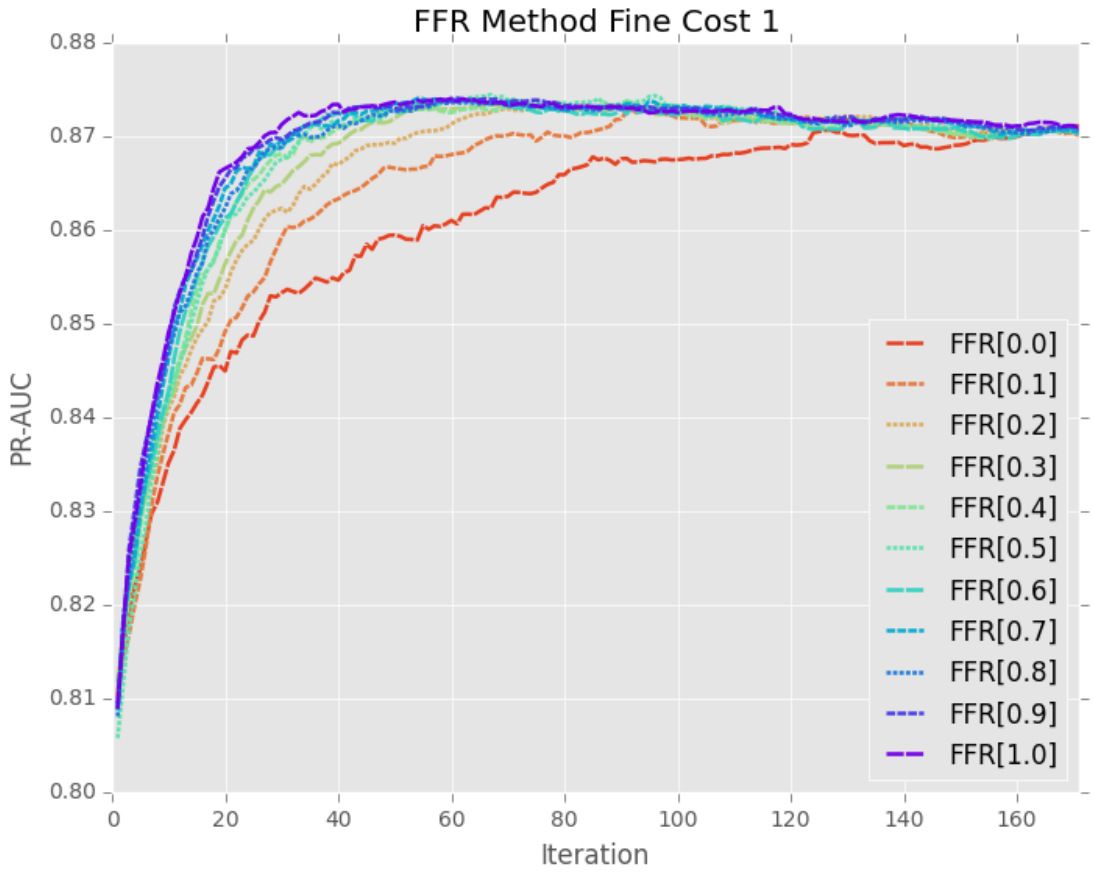

Figure 3.9: The round size is changed to 160 and a fine has a cost of 1. The 0p5 round for instance, corresponds to 0.5 of the total budget being used on fine, so 80 goes to fine and 80 goes to coarse. For the 0p0 round, none of the budget is used for fine and it has the worst performance. After around 0p3 the gains in performance are marginal. The performance increases with the green 1p0 curve outperforming the rest. The results are an average of 10 folds.

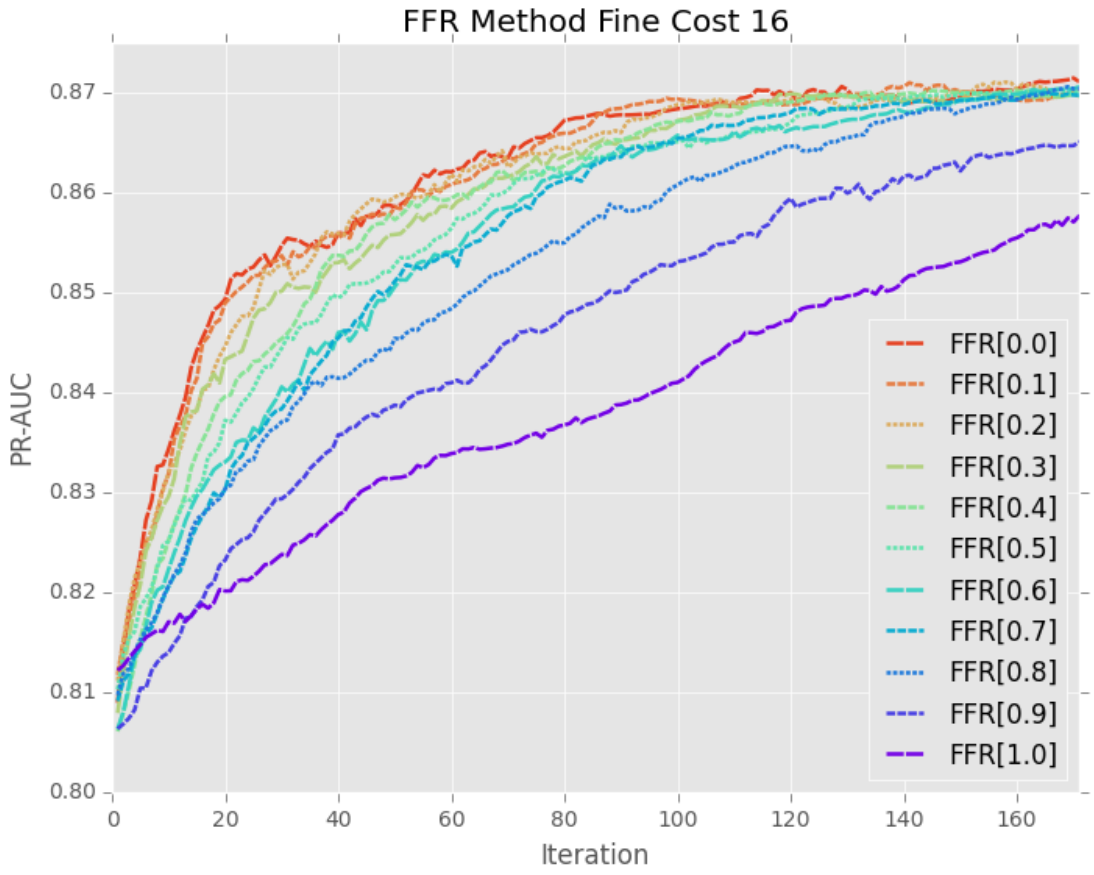

Figure 3.10: The round size is very small only 7.

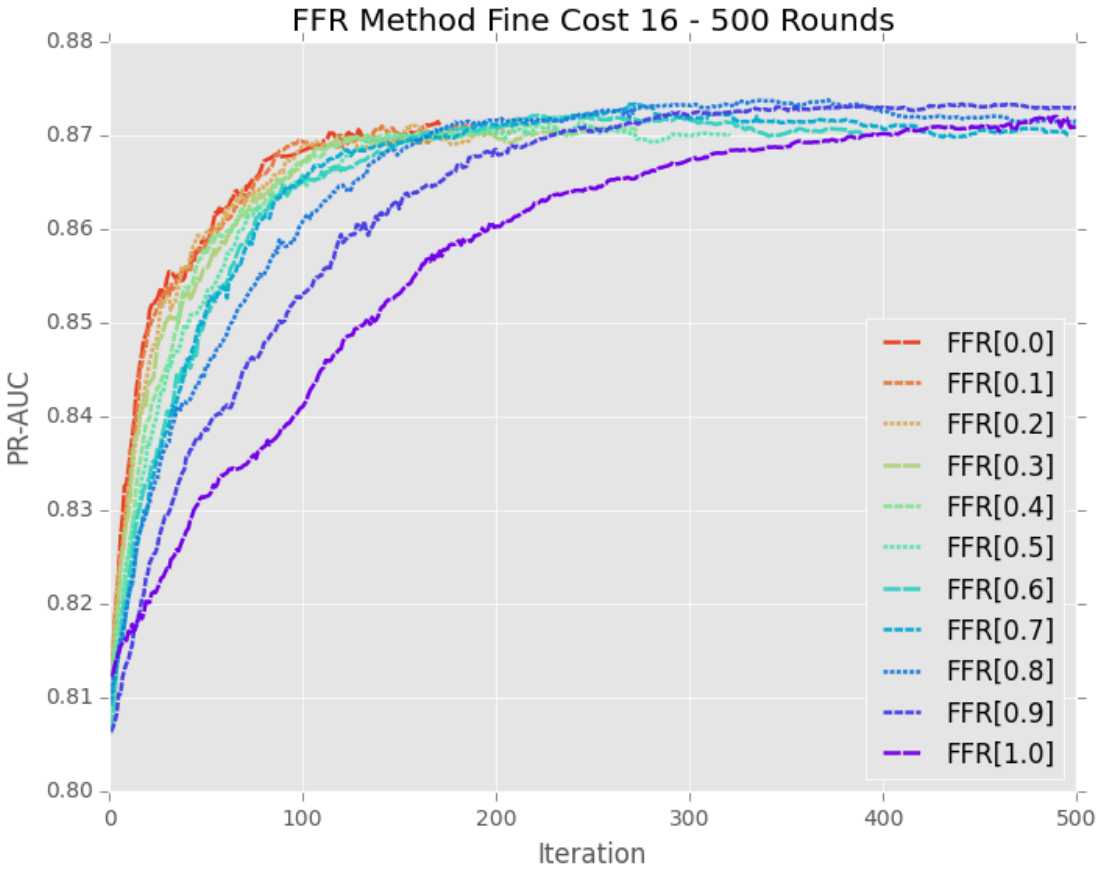

Figure 3.11: The round size is very small only 7. Extended it to 500 rounds.

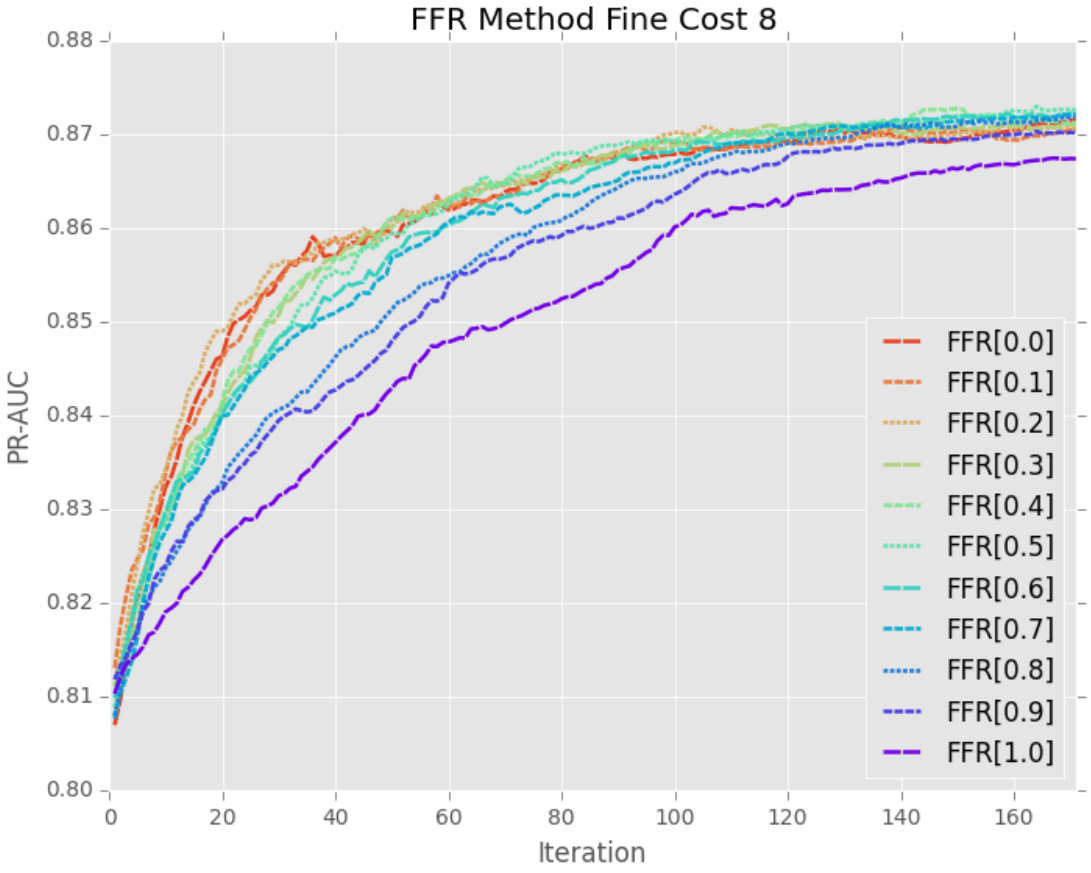

Figure 3.12: The 0.2 is now slightly outperforming the 0.0.

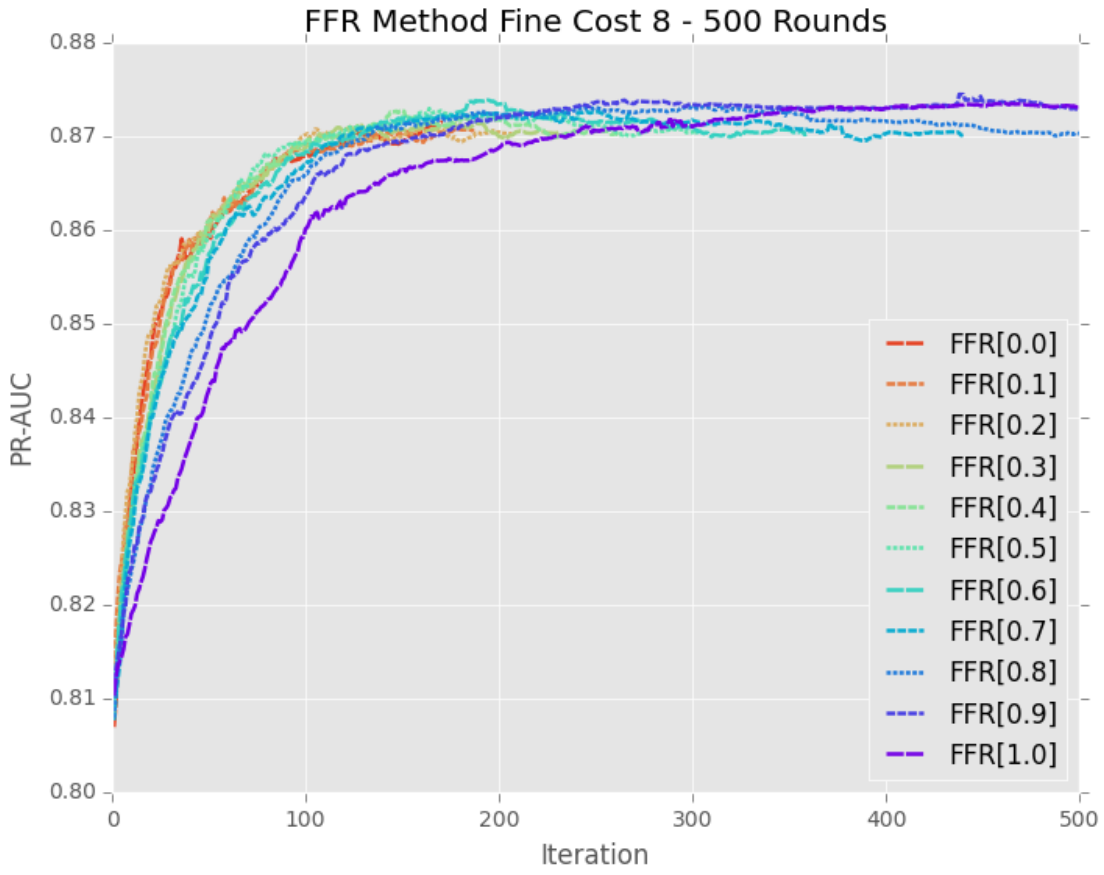

Figure 3.13: The extended picture of the FR cost 8 round size is small only 13.

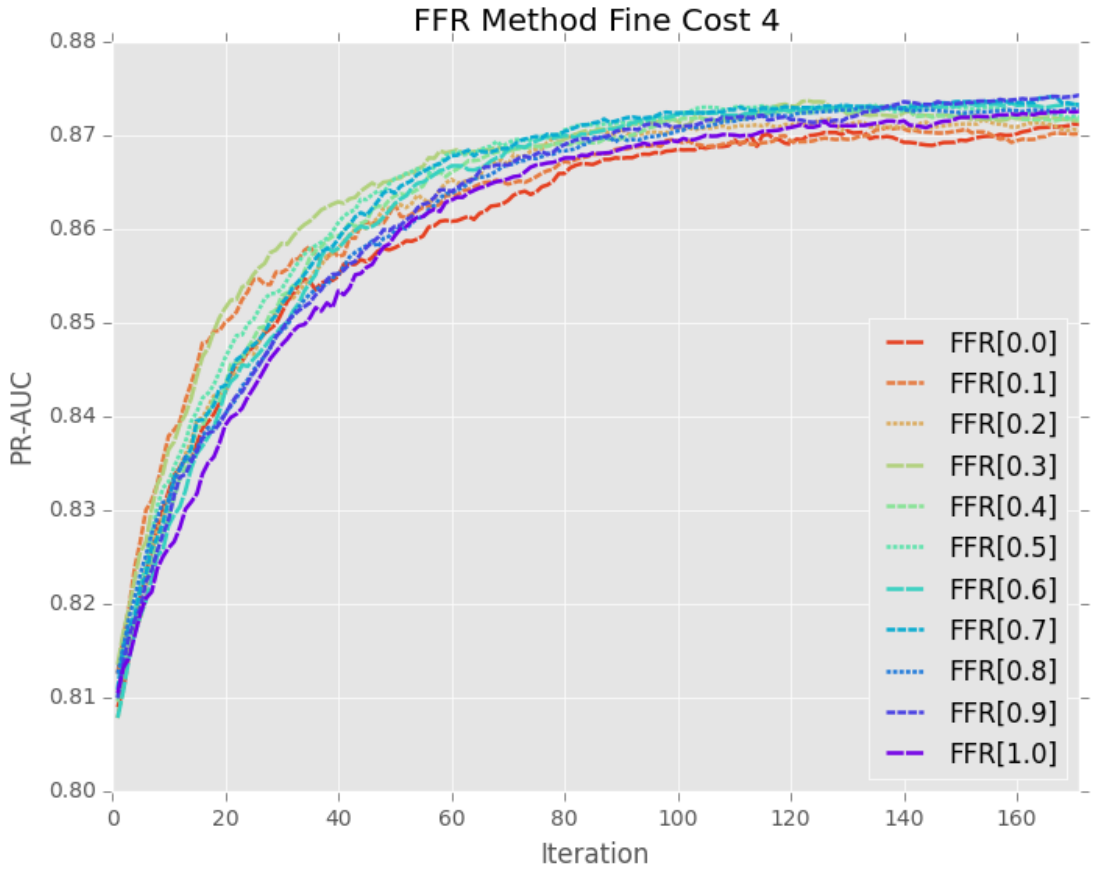

Figure 3.14: The 0.3 is doing the best

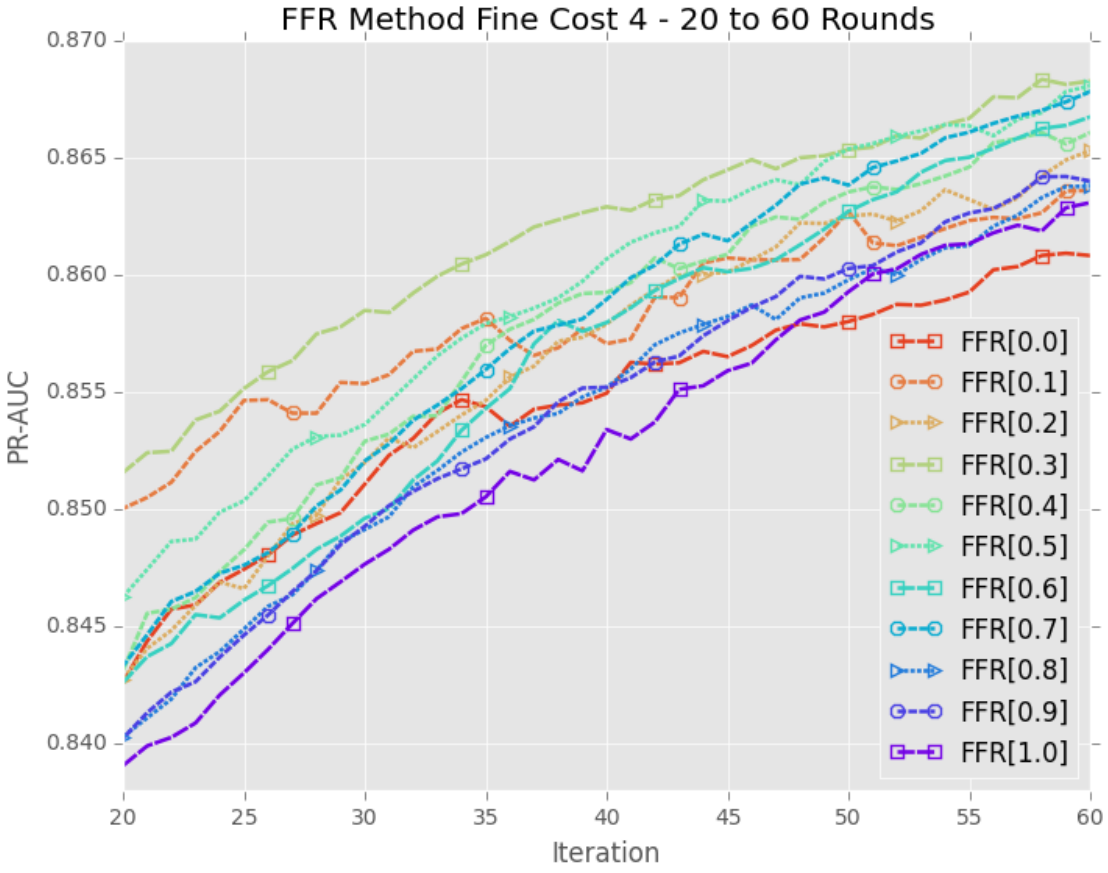

Figure 3.15: The 0.3 is doing the best

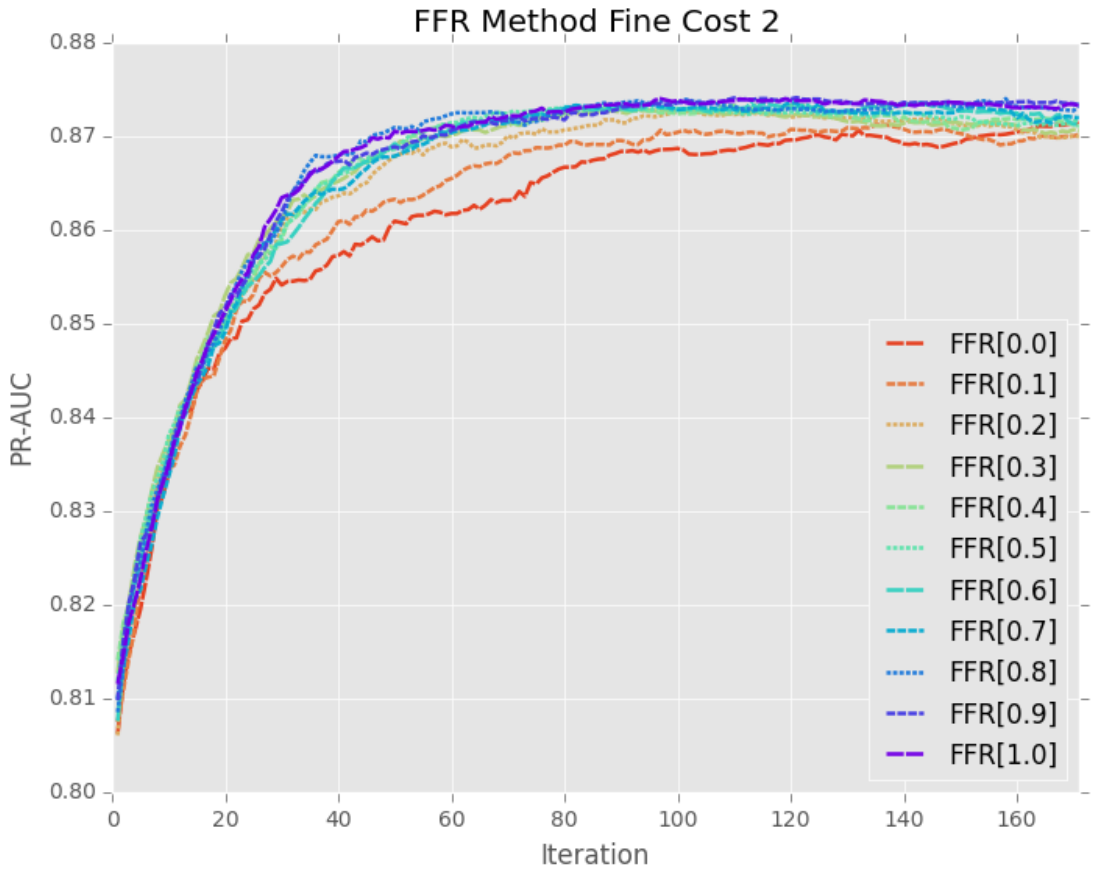

Figure 3.16: The 0.3 is doing the best

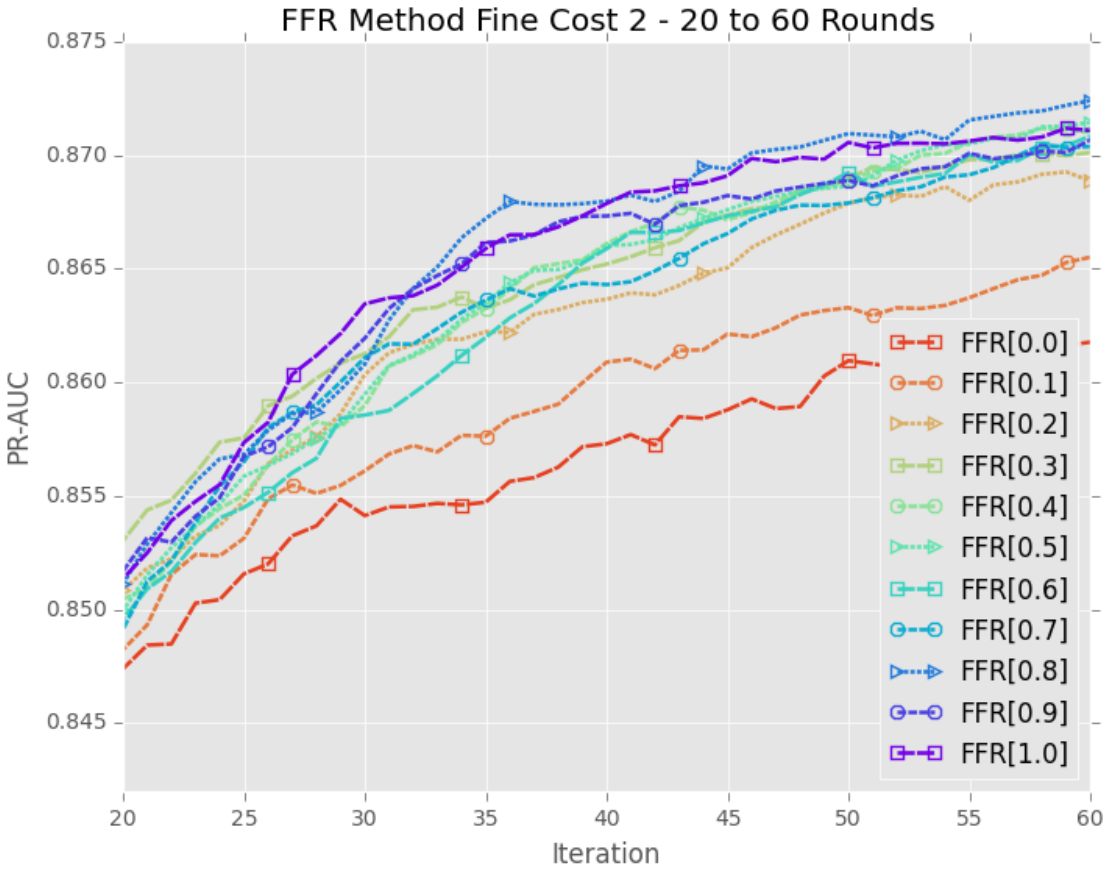

Figure 3.17: The 0.3 is doing the best

Chapter 4

Conclusions and Future Work

I should probably do the Bandit experiments.

Appendix A

Tuning the fine grained classes

All the data and results for tuning the fine grained classes.

Bibliography

[1] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011. 3.1

[2] A. Merialdo, "Improving Collaborative Filtering For New-Users By Smart Object Selection," *In Proceedings of International Conference on Media Features (ICMF)*, May 2001. [Online]. Available: http://www.eurecom.fr/publication/670https://www.eurecom.fr/fr/publication/670/download/mm-kohrar-010508.pdf 3.2

[3] W. Luo, A. Schwing, and R. Urtasun, "Latent structured active learning," in *Advances in Neural Information Processing Systems (NIPS)*, 2013.

[4] S. Dasgupta and D. Hsu, "Hierarchical sampling for active learning," *Proceedings of the 25th international conference on Machine learning - ICML '08*, pp. 208–215, 2008. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1390156.1390183

[5] X. Ling and D. Weld, "Fine-grained entity recognition," *Proceedings of the 26th Conference on Artificial Inteligence*, 2012. [Online]. Available: http://www.cs.washington.edu/ai/pubs/ling-aaai12.pdfhttp://www.aaai.org/ocs/index.php/AAAI/AAAI12/paper/download/5152/5124

[6] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux, "API design for machine learning software: experiences from the scikit-learn project," in *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, 2013, pp. 108–122.