CSE6250 Homework2 Answer

Hongshen Zhao

September 15, 2018

1 Logistic Regression

1.1 Batch Gradient Descent

The negative log-likelihood can be calculated according to

$$NLL(D, \mathbf{w}) = -\sum_{i=1}^{N} \left[(1 - y_i) \log \left(1 - \sigma(\mathbf{w}^T \mathbf{x}_i)\right) + y_i \log \sigma(\mathbf{w}^T \mathbf{x}_i) \right] \tag{1}$$

The maximum likelihood estimator $\mathbf{w}_{\text{MLE}}$ can be found by solving for $\arg\min_{\mathbf{w}} NLL$ through an iterative gradient descent procedure.

a. Derive the gradient of negative log-likelihood in terms of w for this setting.

Suppose

$$\mathbf{w} = \begin{bmatrix} w^{(1)} \\ w^{(2)} \\ \vdots \\ w^{(d)} \end{bmatrix} \tag{2}$$

$$\begin{aligned}
\frac{\partial NLL(D, \mathbf{w})}{\partial w^{(j)}} &= -\sum_{i=1}^{N} \left[(1 - y_i) \frac{-\frac{\partial \sigma(\mathbf{w}^T \mathbf{x}_i)}{\partial w^{(j)}}}{1 - \sigma(\mathbf{w}^T \mathbf{x}_i)} + y_i \frac{\frac{\partial \sigma(\mathbf{w}^T \mathbf{x}_i)}{\partial w^{(j)}}}{\sigma(\mathbf{w}^T \mathbf{x}_i)} \right] \\
&= -\sum_{i=1}^{N} \left[(y_i - 1) \sigma(\mathbf{w}^T \mathbf{x}_i) \frac{\partial \mathbf{w}^T \mathbf{x}_i}{\partial w^{(j)}} + y_i (1 - \sigma(\mathbf{w}^T \mathbf{x}_i)) \frac{\partial \mathbf{w}^T \mathbf{x}_i}{\partial w^{(j)}} \right] \\
&= -\sum_{i=1}^{N} \left[(y_i - 1) \sigma(\mathbf{w}^T \mathbf{x}_i) x_i^{(j)} + y_i (1 - \sigma(\mathbf{w}^T \mathbf{x}_i)) x_i^{(j)} \right] \\
&= -\sum_{i=1}^{N} \left[y_i \sigma(\mathbf{w}^T \mathbf{x}_i) x_i^{(j)} - \sigma(\mathbf{w}^T \mathbf{x}_i) x_i^{(j)} + y_i x_i^{(j)} - y_i \sigma(\mathbf{w}^T \mathbf{x}_i) x_i^{(j)} \right] \\
&= -\sum_{i=1}^{N} \left[y_i - \sigma(\mathbf{w}^T \mathbf{x}_i) \right] x_i^{(j)}
\end{aligned} \tag{3}$$

$$\begin{aligned}
\nabla NLL(D, \mathbf{w}) &= \begin{bmatrix} -\sum_{i=1}^{N} \left[y_i - \sigma(\mathbf{w}^T \mathbf{x}_i) \right] x_i^{(1)} \\ -\sum_{i=1}^{N} \left[y_i - \sigma(\mathbf{w}^T \mathbf{x}_i) \right] x_i^{(2)} \\ \vdots \\ -\sum_{i=1}^{N} \left[y_i - \sigma(\mathbf{w}^T \mathbf{x}_i) \right] x_i^{(d)} \end{bmatrix} \\
&= -\sum_{i=1}^{N} \left[y_i - \sigma(\mathbf{w}^T \mathbf{x}_i) \right] \mathbf{x}_i
\end{aligned} \tag{4}$$

1.2 Stochastic Gradient Descent

If N and d are very large, it may be prohibitively expensive to consider every patient in D before applying an update to w. One alternative is to consider stochastic gradient descent, in which an update is applied after only considering a single patient.

a. Show the log likelihood, l, of a single $(\mathbf{x}_t, y_t)$ pair.

$$l = (1 - y_t) \log (1 - \sigma(\mathbf{w}^T \mathbf{x}_t)) + y_t \log \sigma(\mathbf{w}^T \mathbf{x}_t) \tag{5}$$

b. Show how to update the coefficient vector $\mathbf{w}_t$ when you get a patient feature vector $\mathbf{x}_t$ and physician feedback label y_t at time t using $\mathbf{w}_{t-1}$ (assume learning rate η is given).

$$\mathbf{w}_t \leftarrow \mathbf{w}_{t-1} + \eta \nabla l(\mathbf{w}_{t-1}) \tag{6}$$

$$\begin{aligned} \mathbf{w}_t &= \mathbf{w}_{t-1} + \eta \begin{bmatrix} [y_t - \sigma(\mathbf{w}_{t-1}^T \mathbf{x}_t)] x_t^{(1)} \\ [y_t - \sigma(\mathbf{w}_{t-1}^T \mathbf{x}_t)] x_t^{(2)} \\ \vdots \\ [y_t - \sigma(\mathbf{w}_{t-1}^T \mathbf{x}_t)] x_t^{(d)} \end{bmatrix} \\ &= \mathbf{w}_{t-1} + \eta [y_t - \sigma(\mathbf{w}_{t-1}^T \mathbf{x}_t)] \mathbf{x}_t \end{aligned} \tag{7}$$

c. What is the time complexity of the update rule from b if $\mathbf{x}_t$ is very sparse?

If the data is very sparse, only a very small (constant on average) number of elements in $\mathbf{x}_t$ will be involved in the calculation of $\sigma(\mathbf{w}_{t-1}^T \mathbf{x}_t)$ and $[y_t - \sigma(\mathbf{w}_{t-1}^T \mathbf{x}_t)]\mathbf{x}_t$. This corresponds to the implementation in `lrsgd.py` shown below. So, the time complexity is $O(1)$.

```
y_estimated=1.0/(1.0+math.exp(-math.fsum((self.weight[f]*v for f, v in X))))
for f,v in X:
    self.weight[f]=self.weight[f]+self.eta*(y-y_estimated)*v
```

d. Briefly explain the consequence of using a very large η and very small η.

Very large η: The model will not converge to the optimal point but jump around instead. Large η may also cause the model to diverge because the update step is too large and it keeps jumping over the optimal point.

Very small η: Can cause the model to take forever to converge to a local optimal point because the update step is so small. Can be easily stuck at a local saddle point and lose the opportunity to search for the global optimal point.

e. Show how to update $\mathbf{w}_t$ under the penalty of L2 norm regularization. In other words, update $\mathbf{w}_t$ according to $l - \mu||\mathbf{w}||_2^2$, where μ is a constant. What's the time complexity?

$$\frac{\partial (l - \mu||\mathbf{w}_{t-1}||_2^2)}{\partial \mathbf{w}_{t-1}} = \frac{\partial l}{\partial \mathbf{w}_{t-1}} - 2\mu \mathbf{w}_{t-1} \tag{8}$$

$$\mathbf{w}_t \leftarrow \mathbf{w}_{t-1} + \eta \nabla \left[l(\mathbf{w}_{t-1}) - \mu||\mathbf{w}_{t-1}||_2^2 \right] \tag{9}$$

$$\begin{aligned} \mathbf{w}_t &= \mathbf{w}_{t-1} + \eta \begin{bmatrix} [y_t - \sigma(\mathbf{w}_{t-1}^T \mathbf{x}_t)] x_t^{(1)} \\ [y_t - \sigma(\mathbf{w}_{t-1}^T \mathbf{x}_t)] x_t^{(2)} \\ \vdots \\ [y_t - \sigma(\mathbf{w}_{t-1}^T \mathbf{x}_t)] x_t^{(d)} \end{bmatrix} - 2\mu\eta \mathbf{w}_{t-1} \\ &= (1 - 2\mu\eta)\mathbf{w}_{t-1} + \eta [y_t - \sigma(\mathbf{w}_{t-1}^T \mathbf{x}_t)] \mathbf{x}_t \end{aligned} \tag{10}$$

Because the vector $\mathbf{w_{t-1}}$ always needs to be scaled by $(1 - 2\mu\eta)$ which has a dimension d, the time complexity is now $O(d)$ per iteration even if $\mathbf{x}_t$ is sparse.

```
y_estimated=1.0/(1.0+math.exp(-math.fsum((self.weight[f]*v for f, v in X))))
self.weight=[(1-2*self.eta*self.mu)*v for v in self.weight]
for f,v in X:
    self.weight[f]=self.weight[f]+self.eta*(y-y_estimated)*v
```

2 Programming

2.1 Descriptive Statistics

Metrics	Deceased patients	Alive patients
Event Count		
1. Average Event Count	1027.7385	683.1553
2. Max Event Count	16829	12627
3. Min Event Count	2	1
Encounter Count		
1. Average Encounter Count	24.8393	18.6955
2. Max Encounter Count	375	391
3. Min Encounter Count	1	1
Record Length		
1. Average Record Length	157.0419	194.7028
2. Median Record Length	25	16
3. Max Record Length	5364	3103
4. Min Record Length	0	0
Common Diagnosis		
	DIAG320128, 416	DIAG320128, 1018
	DIAG319835, 413	DIAG319835, 721
	DIAG313217, 377	DIAG317576, 719
	DIAG197320, 346	DIAG42872402, 674
	DIAG132797, 297	DIAG313217, 641
Common Laboratory		
	LAB3009542, 32765	LAB3009542, 66937
	LAB3023103, 28395	LAB3000963, 57751
	LAB3000963, 28308	LAB3023103, 57022
	LAB3018572, 27383	LAB3018572, 54721
	LAB3016723, 27060	LAB3007461, 53560
Common Medication		
	DRUG19095164, 6396	DRUG19095164, 12468
	DRUG43012825, 5451	DRUG43012825, 10389
	DRUG19049105, 4326	DRUG19049105, 9351
	DRUG956874, 3962	DRUG19122121, 7586
	DRUG19122121, 3910	DRUG956874, 7301

2.2 Transform data

No report deliverable in this section.

2.3 SGD Logistic Regression

b. Show the ROC curve generated by `test.py` in this writing report for different learning rates η and regularization parameters μ combination and briefly explain the result.

I performed a grid search with η in 0.001, 0.01, 0.05, 0.1, 0.4, and 0.8, μ in 0, 0.00001, 0.0001, 0.001, 0.01, and 0.1. The ROC curves are shown in Figure 1, Figure 2, Figure 3, Figure 4, Figure 5, and Figure 6. Each Figure plots the ROC curve with a fixed μ value and various η values.

For the regularization strength μ, increasing the value only made the ROC curve area smaller if the learning rate η remaining the same. I think the reason was because of the sparsity nature of the $\mathbf{x}_t$ vectors. With L2 regularization, regardless of the sparsity of $\mathbf{x}_t$, I was scaling down the entire weight vector $\mathbf{w}_{t-1}$ by $(1 - 2\mu\eta)$. If the $\mathbf{x}_t$ vectors were very sparse and the most important features were not commonly shared, those weights would get scaled down too much by the L2 norm updates, before the corresponding dimensions in the weight vector got the infrequent updates from the native log-likelihood derivative part, which could cause the performance to be worse and worse and the model only

diverged. So, for this dataset, L2 norm regularization was not a good choice to reduce overfitting as it caused underfitting.

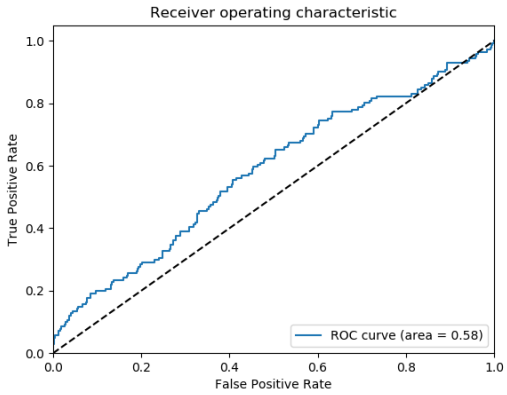
For the learning rate η, the best values I found was 0.05 and 0.1 in my grid search, the true optimal value could be between 0.01 and 0.4 which could be found using more fine-grained grid search. The highes ROC curve areas I got was 0.65. With smaller learning rates, the ROC areas were smaller because the model failed to converge to the optimal point within the epoch which resulted in degraded performance. With larger learning rates, the ROC areas were also smaller because the step of gradient descent updates were too large which caused the model to jump around the optimal point but never converged or it might even diverge.

2.4 Hadoop

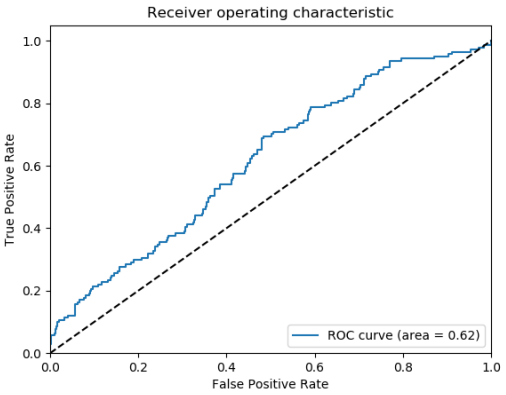
c. Compare the performance with that of previous problem and briefly analyze why the difference.

I found that the optimal learning rate changed after using ensemble with sampling ratio $r < 1$. By sampling the training data, the number of steps of update in the weights for the classifiers was fewer compared to the previous section where I used the entire data set to train a model in one epoch. So, the optimal learning rate became larger to compensate the effect of fewer update steps. The optimal learning rate I fond was 0.8 by performing grid search on η among 0.05, 0.1, 0.4, and 0.8. The ROC curves were shown in Figure 7, Figure 8, Figure 9, and Figure 10. With a sampling ratio of 0.4, the effective learning rate was $r \times \eta = 0.32$ when compared to the previous section. By varying the number of models, the highest ROC area I got was 0.69 with 50 and 100 models. With more models used in ensemble, the variance was reduced and the testing performance was improved.

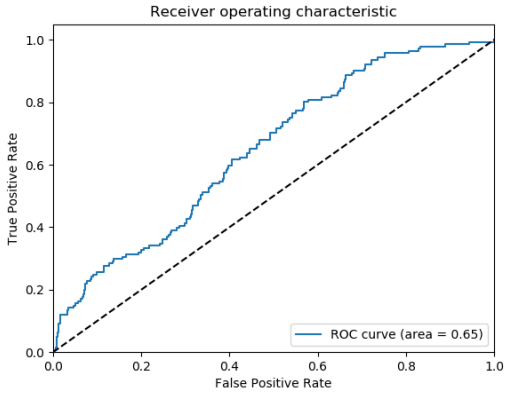
I further experimented the effect of sampling ratio. By fixing η at 0.8 and model number n at 100. The ROC curves with different sampling ratios was shwon in Figure 11. The observation was higher sampling ratios degraded the performance of the ensembled model. There were two reasons. First, the higher the sampling ratio, the higher the effective learning rate. By referring to the previous section **2.3**, $\eta = 0.8$ was too large to be trained with the entire training data set, so it would give worse performance. Second, using $r = 1$ meaned all the classifiers were trained with exactly the same data, which reduced the effect of ensemble because all the classifiers could converge to about the same. With a sampling ratio of 1, it did not help to reduce variance if no other configuration variance for the classifiers. If the sampling ratio was too small the ensemble model would underfit the data which give worse performance.



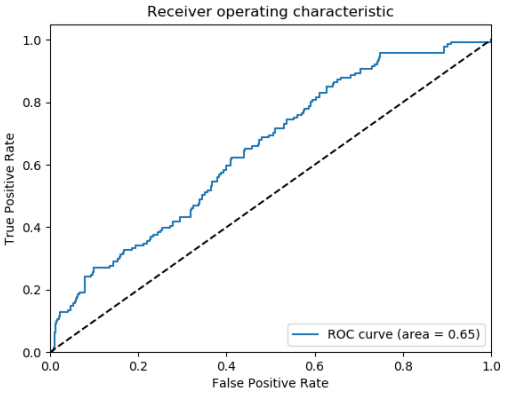
(a) $\eta = 0.001$



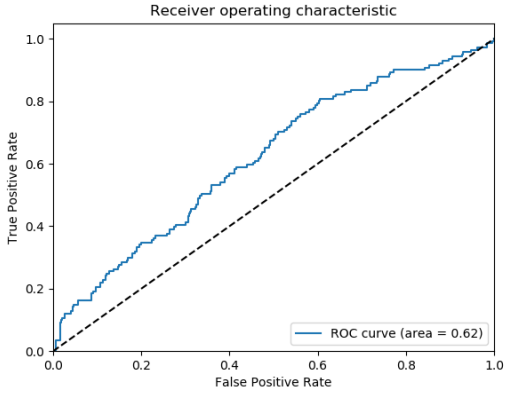
(b) $\eta = 0.01$



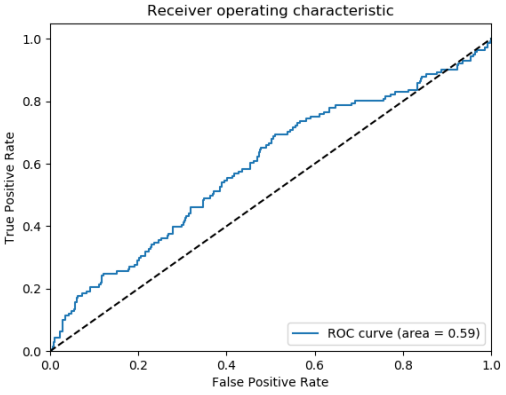
(c) $\eta = 0.05$



(d) $\eta = 0.1$

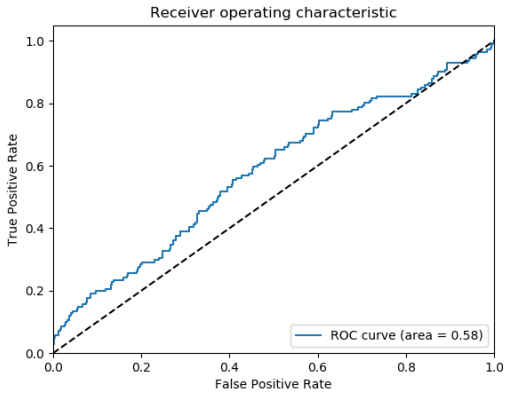


(e) $\eta = 0.4$

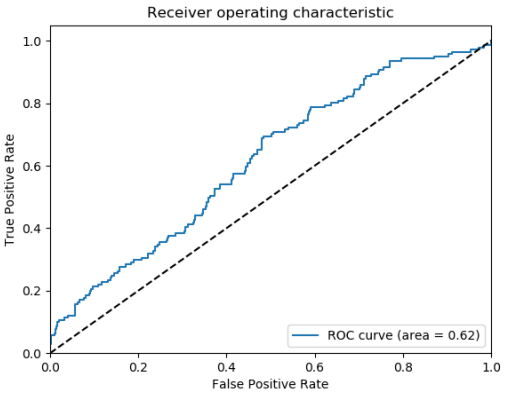


(f) $\eta = 0.8$

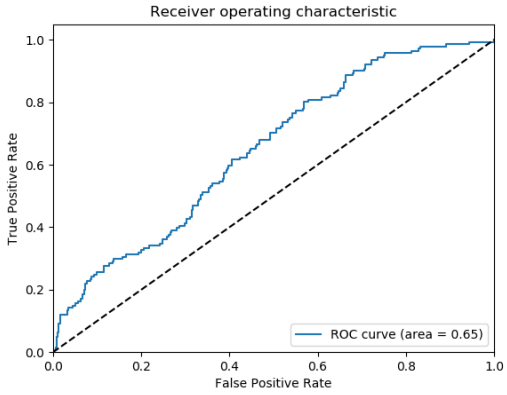
Figure 1: SGD Logistic Regression ROC curves with $\mu = 0$



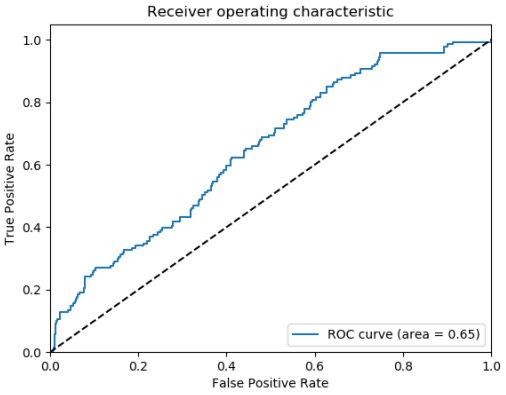
(a) $\eta = 0.001$



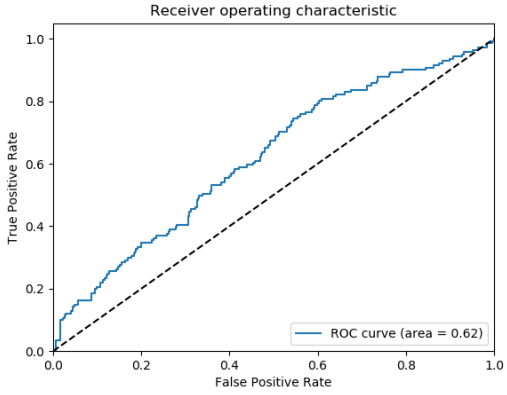
(b) $\eta = 0.01$



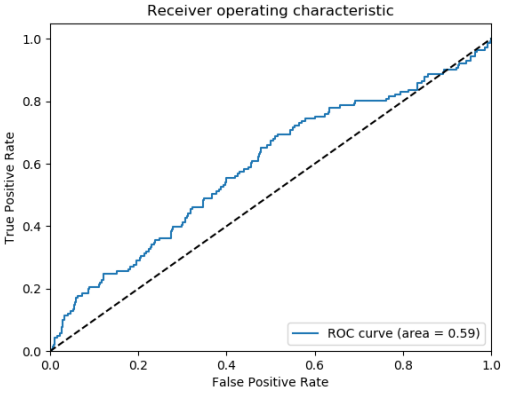
(c) $\eta = 0.05$



(d) $\eta = 0.1$

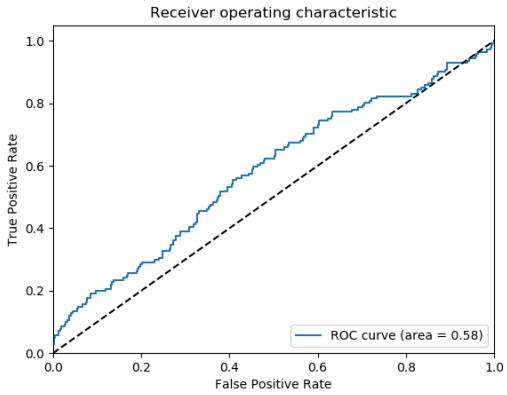


(e) $\eta = 0.4$

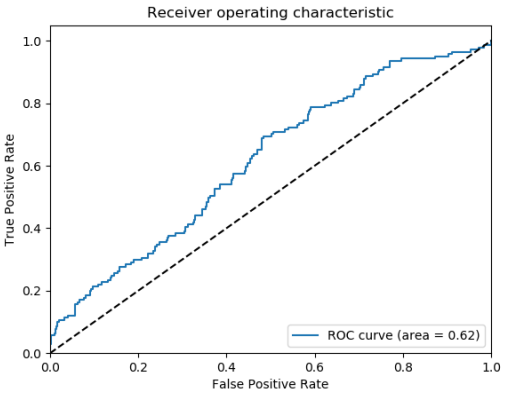


(f) $\eta = 0.8$

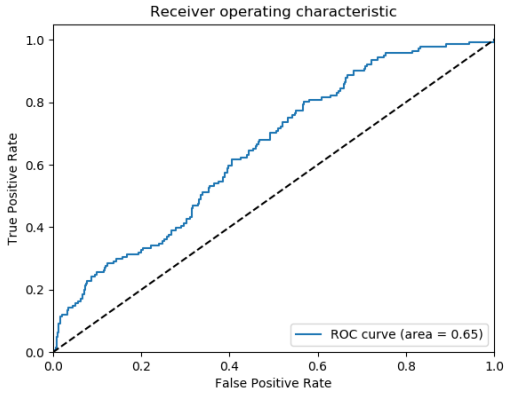
Figure 2: SGD Logistic Regression ROC curves with $\mu = 0.00001$



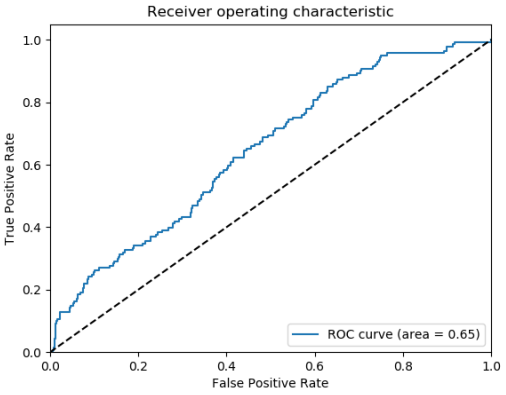
(a) $\eta = 0.001$



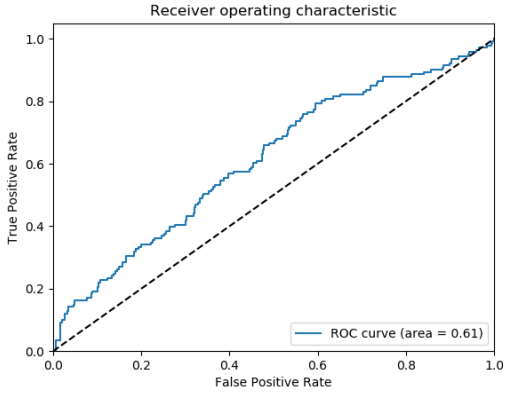
(b) $\eta = 0.01$



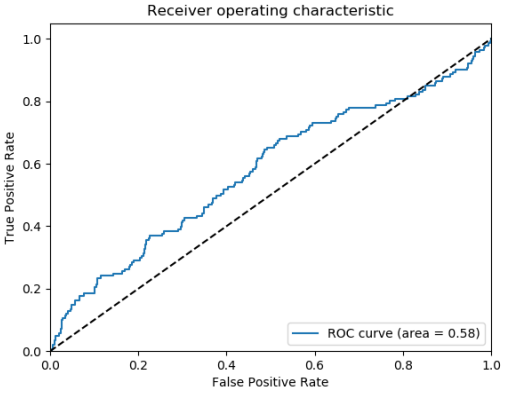
(c) $\eta = 0.05$



(d) $\eta = 0.1$

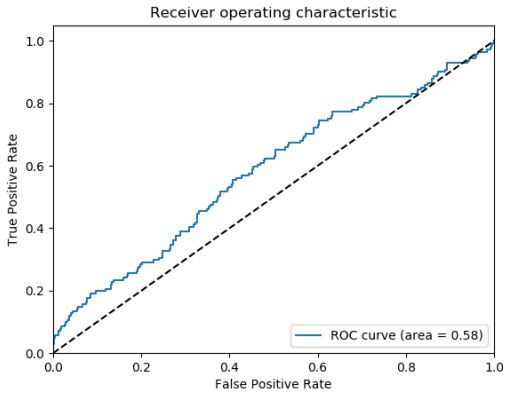


(e) $\eta = 0.4$

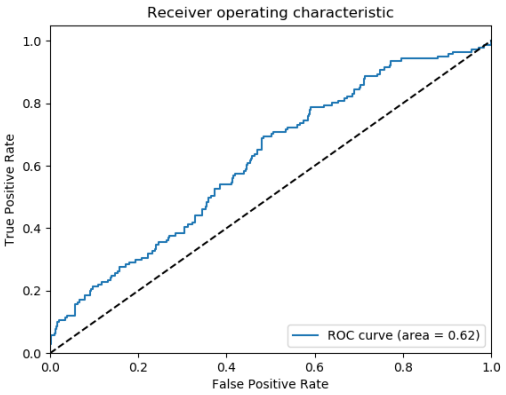


(f) $\eta = 0.8$

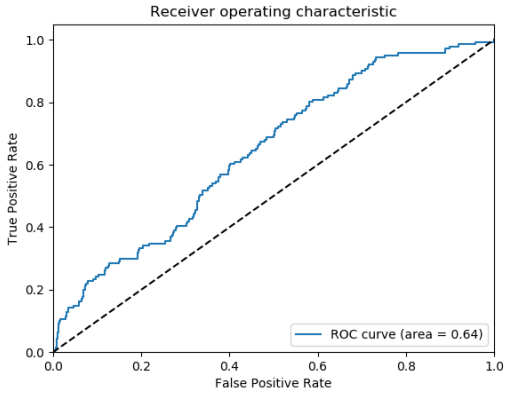
Figure 3: SGD Logistic Regression ROC curves with $\mu = 0.0001$



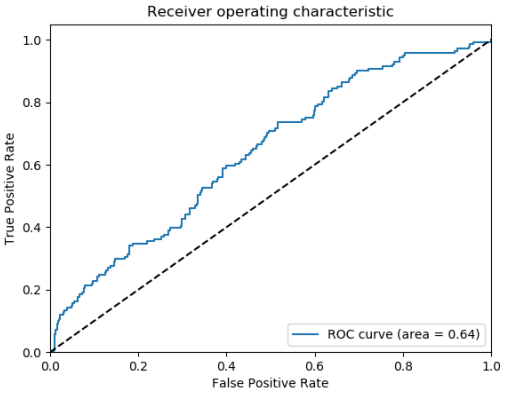
(a) $\eta = 0.001$



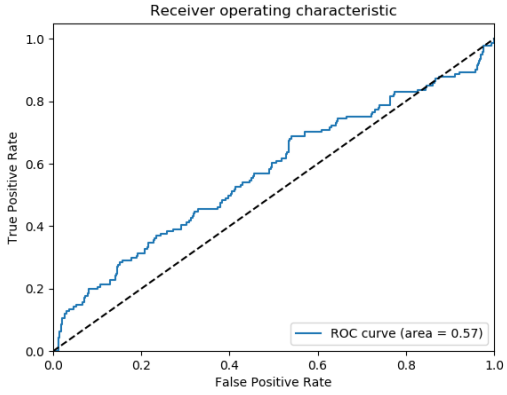
(b) $\eta = 0.01$



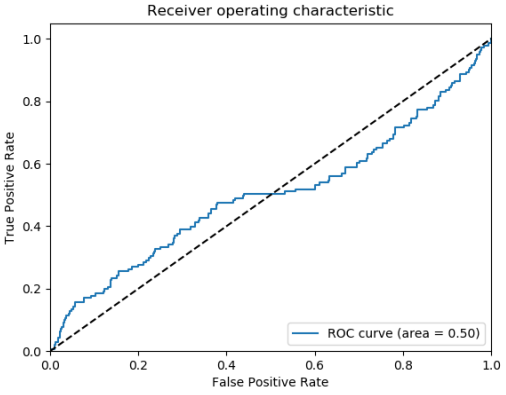
(c) $\eta = 0.05$



(d) $\eta = 0.1$

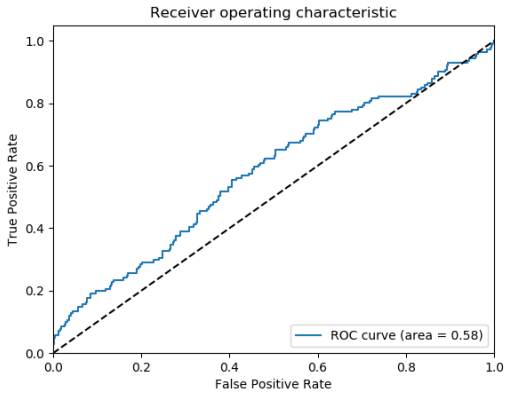


(e) $\eta = 0.4$

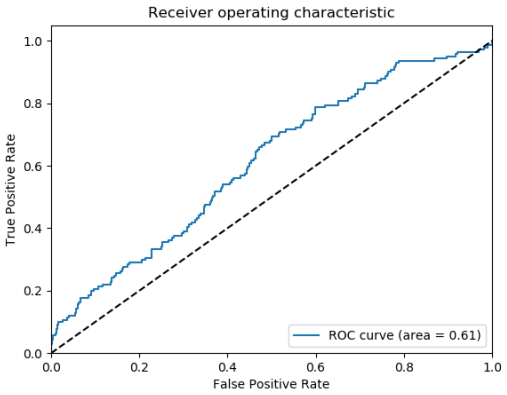


(f) $\eta = 0.8$

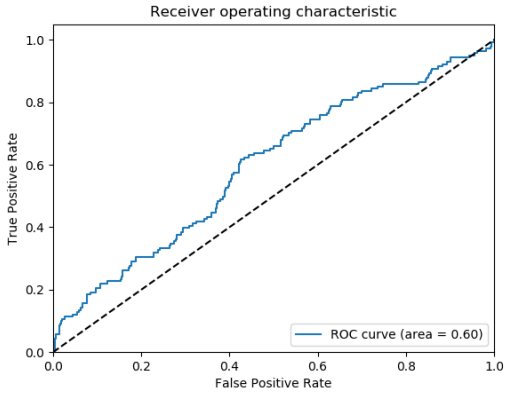
Figure 4: SGD Logistic Regression ROC curves with $\mu = 0.001$



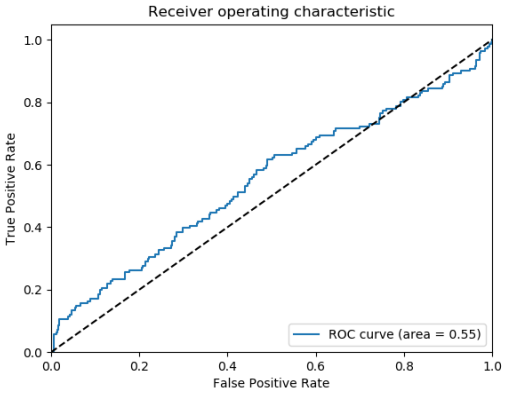
(a) $\eta = 0.001$



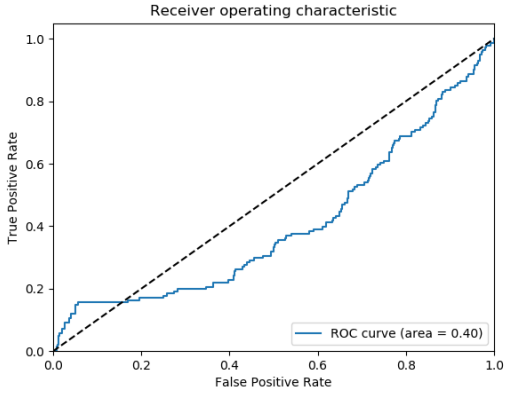
(b) $\eta = 0.01$



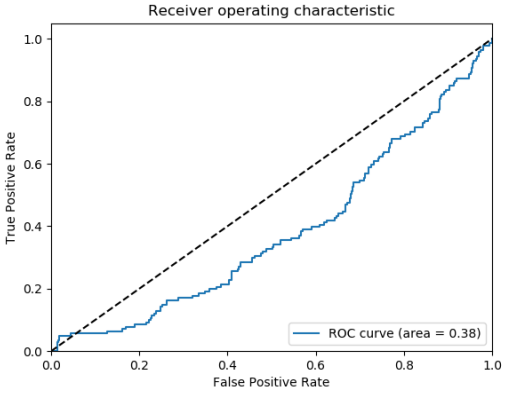
(c) $\eta = 0.05$



(d) $\eta = 0.1$

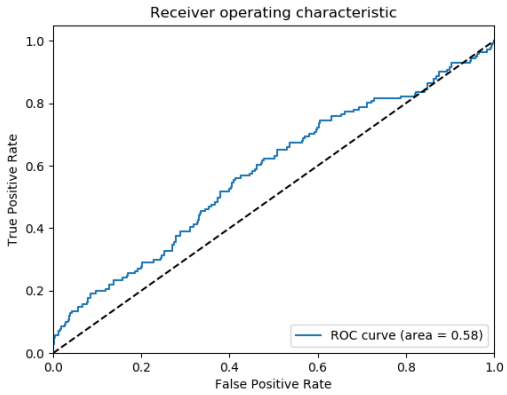


(e) $\eta = 0.4$

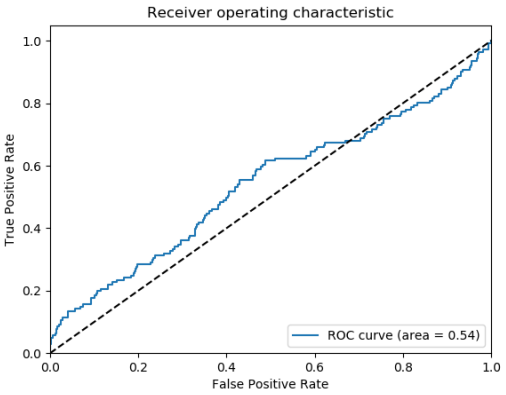


(f) $\eta = 0.8$

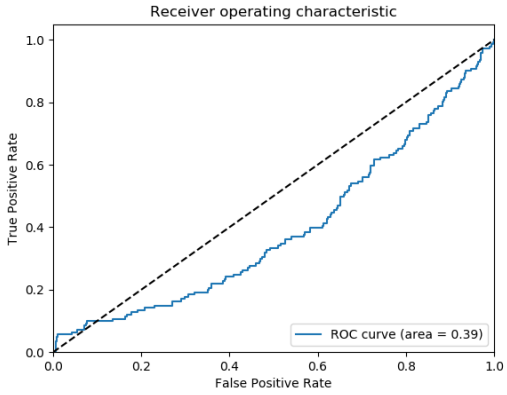
Figure 5: SGD Logistic Regression ROC curves with $\mu = 0.01$



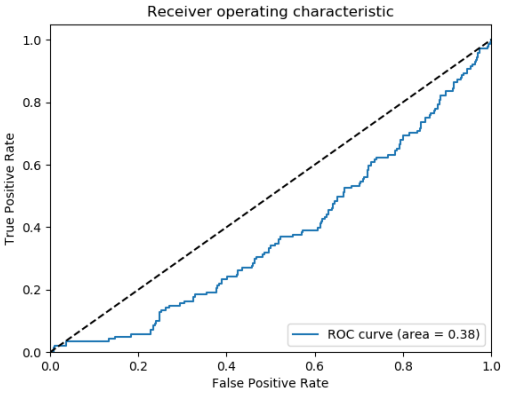
(a) $\eta = 0.001$



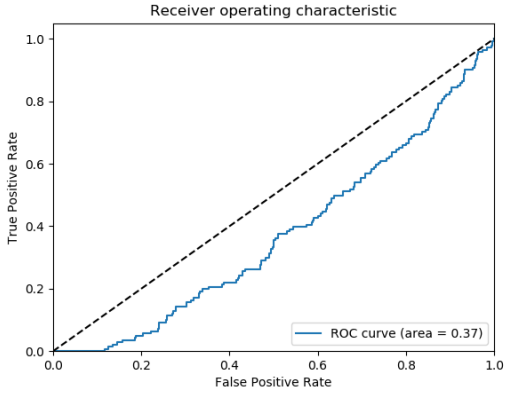
(b) $\eta = 0.01$



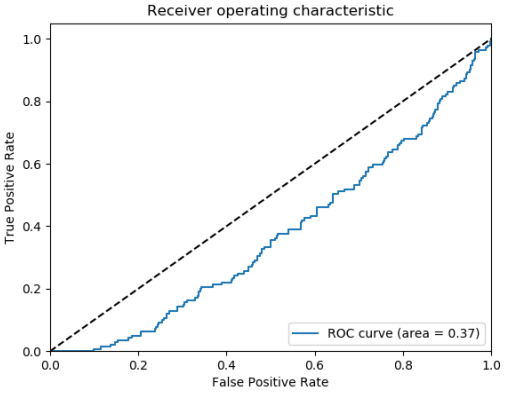
(c) $\eta = 0.05$



(d) $\eta = 0.1$

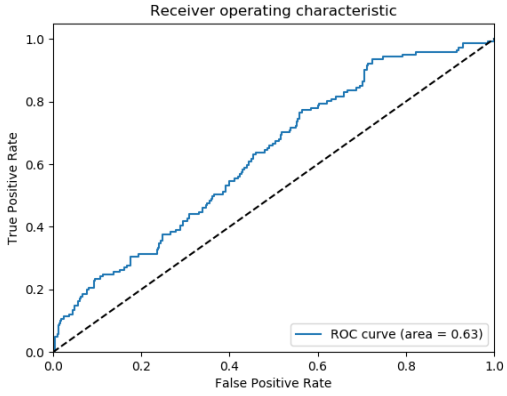


(e) $\eta = 0.4$

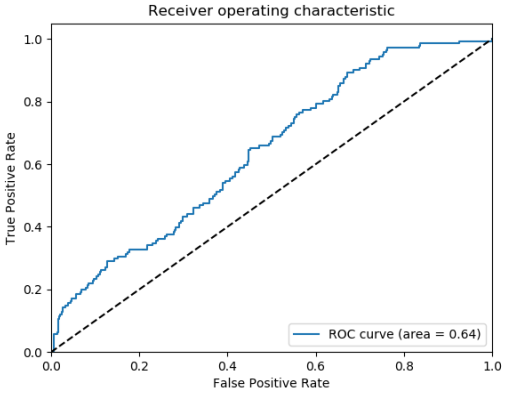


(f) $\eta = 0.8$

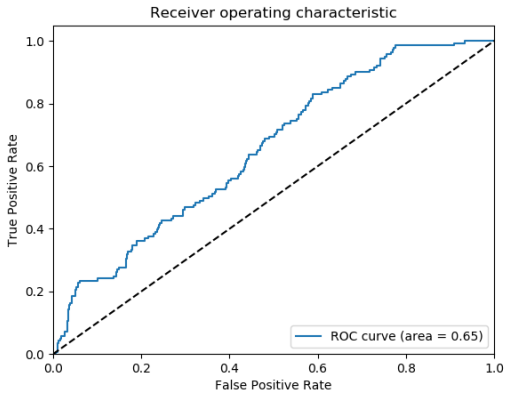
Figure 6: SGD Logistic Regression ROC curves with $\mu = 0.1$



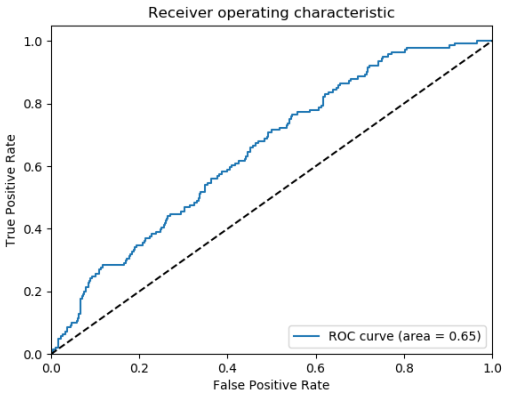
(a) $\eta = 0.05$



(b) $\eta = 0.1$

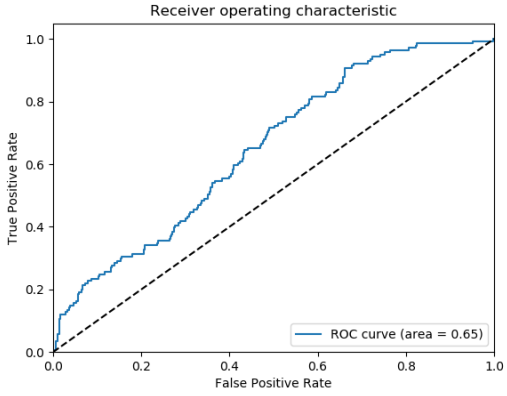


(c) $\eta = 0.4$

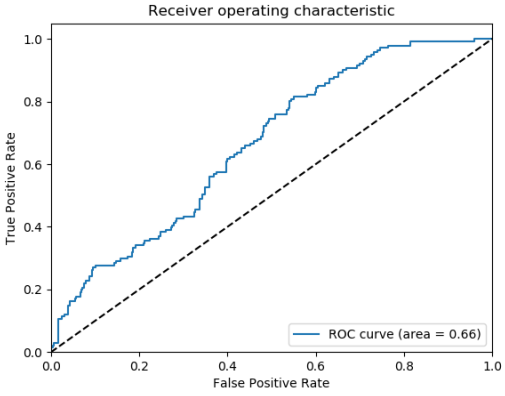


(d) $\eta = 0.8$

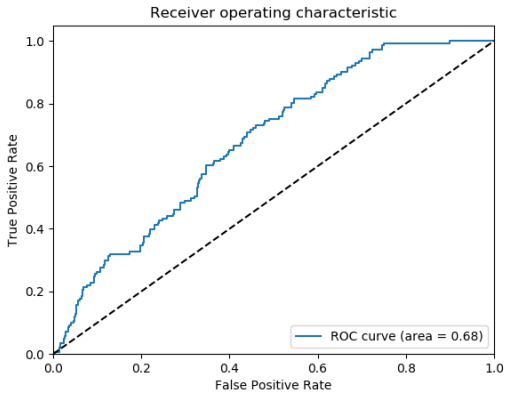
Figure 7: Ensemble with MapReduce ROC curves with $\mu = 0, r = 0.4, n = 5$



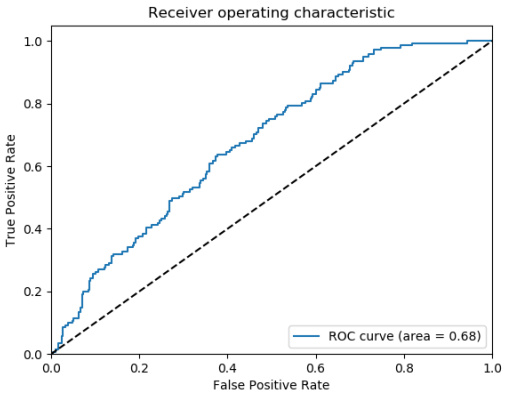
(a) $\eta = 0.05$



(b) $\eta = 0.1$

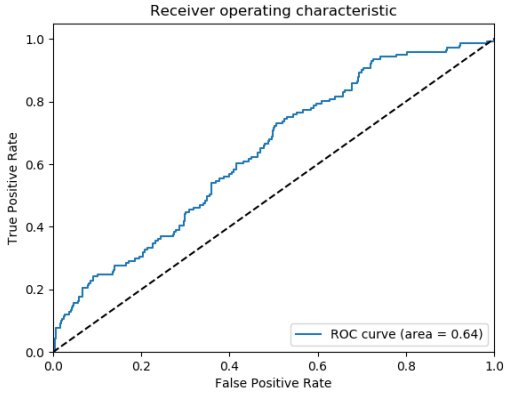


(c) $\eta = 0.4$

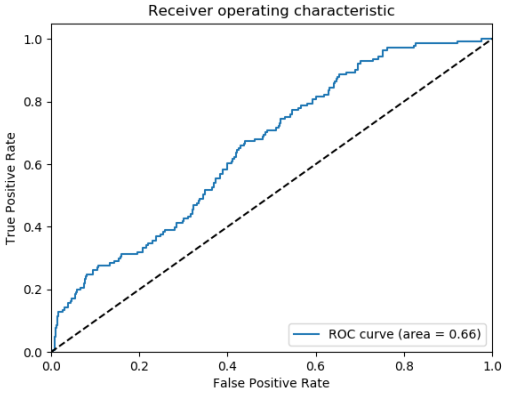


(d) $\eta = 0.8$

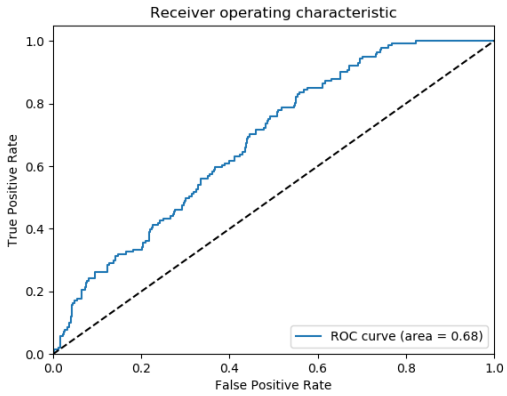
Figure 8: Ensemble with MapReduce ROC curves with $\mu = 0, r = 0.4, n = 25$



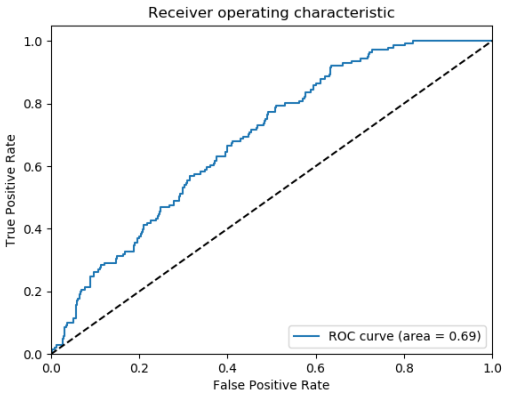
(a) $\eta = 0.05$



(b) $\eta = 0.1$

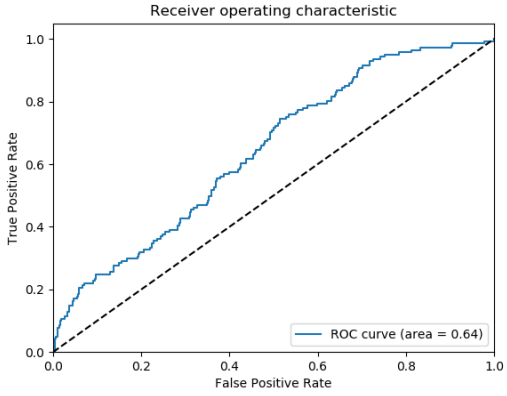


(c) $\eta = 0.4$

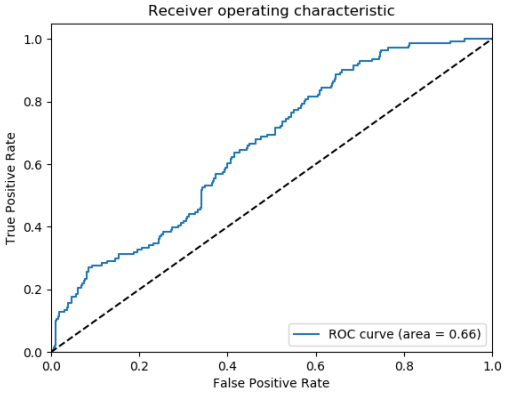


(d) $\eta = 0.8$

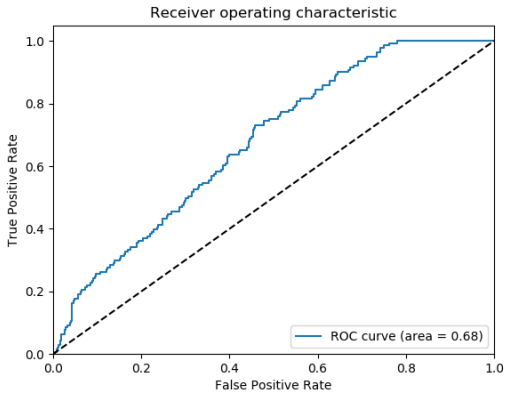
Figure 9: Ensemble with MapReduce ROC curves with $\mu = 0, r = 0.4, n = 50$



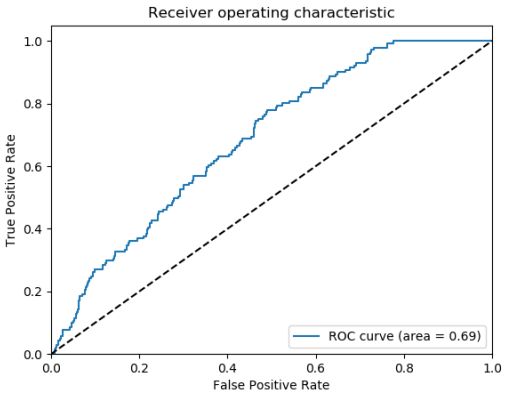
(a) $\eta = 0.05$



(b) $\eta = 0.1$

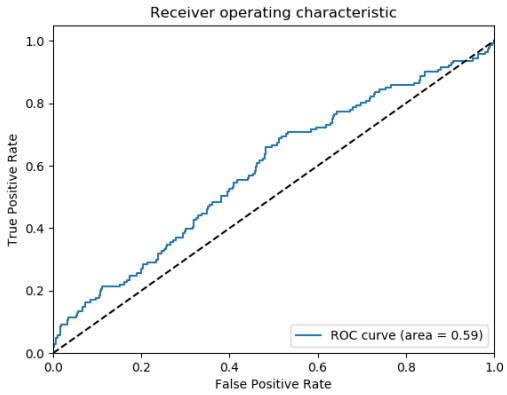


(c) $\eta = 0.4$

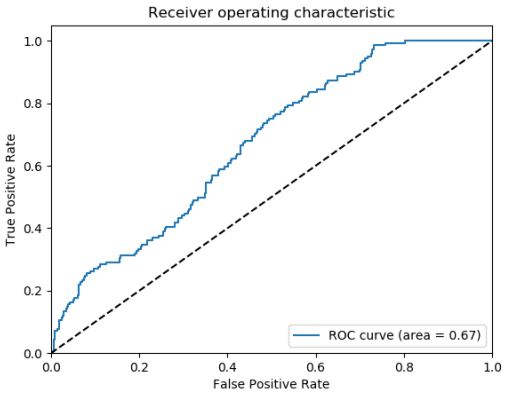


(d) $\eta = 0.8$

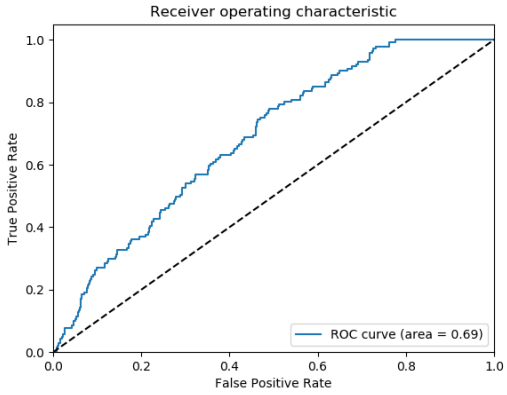
Figure 10: Ensemble with MapReduce ROC curves with $\mu = 0, r = 0.4, n = 100$



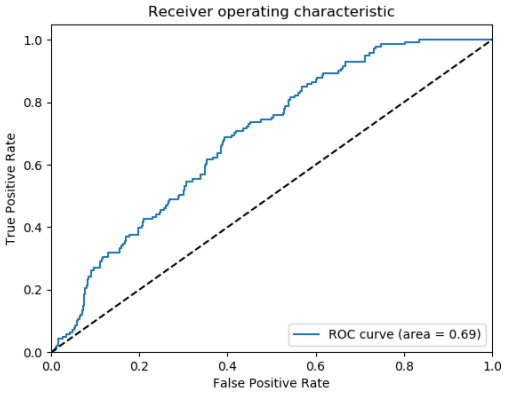
(a) $r = 0.005$



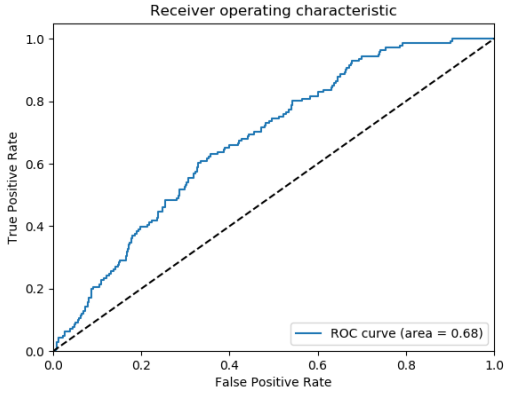
(b) $r = 0.1$



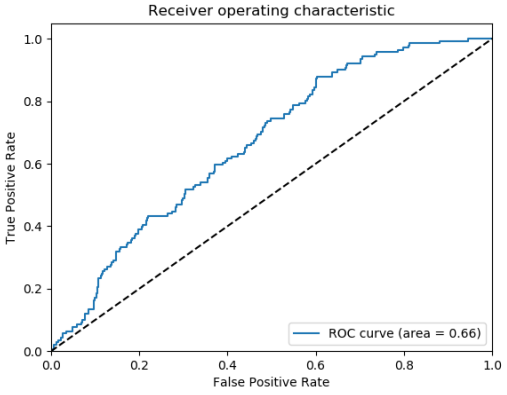
(c) $r = 0.4$



(d) $r = 0.6$



(e) $r = 0.8$



(f) $r = 1$

Figure 11: Ensemble with MapReduce ROC curves with $\mu = 0, \eta = 0.8, n = 100$