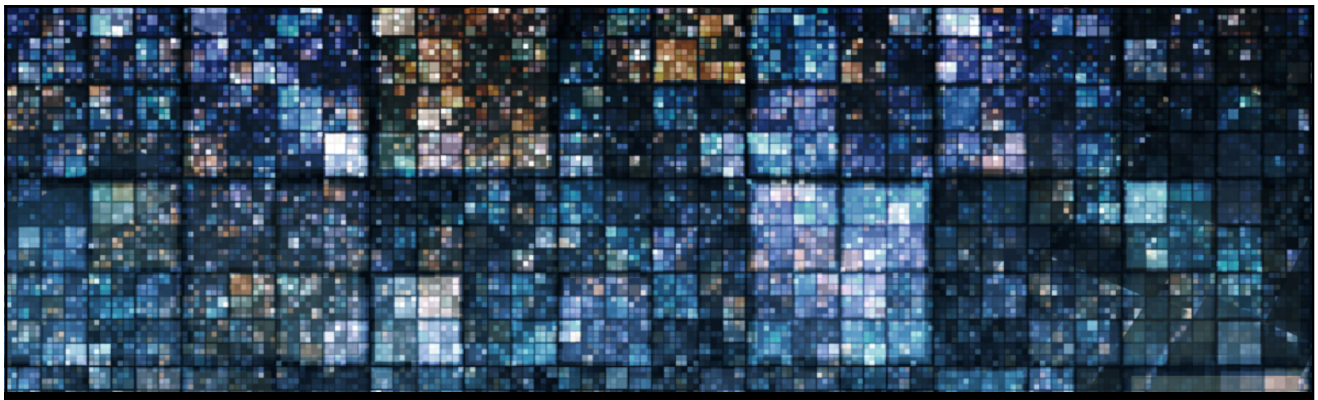
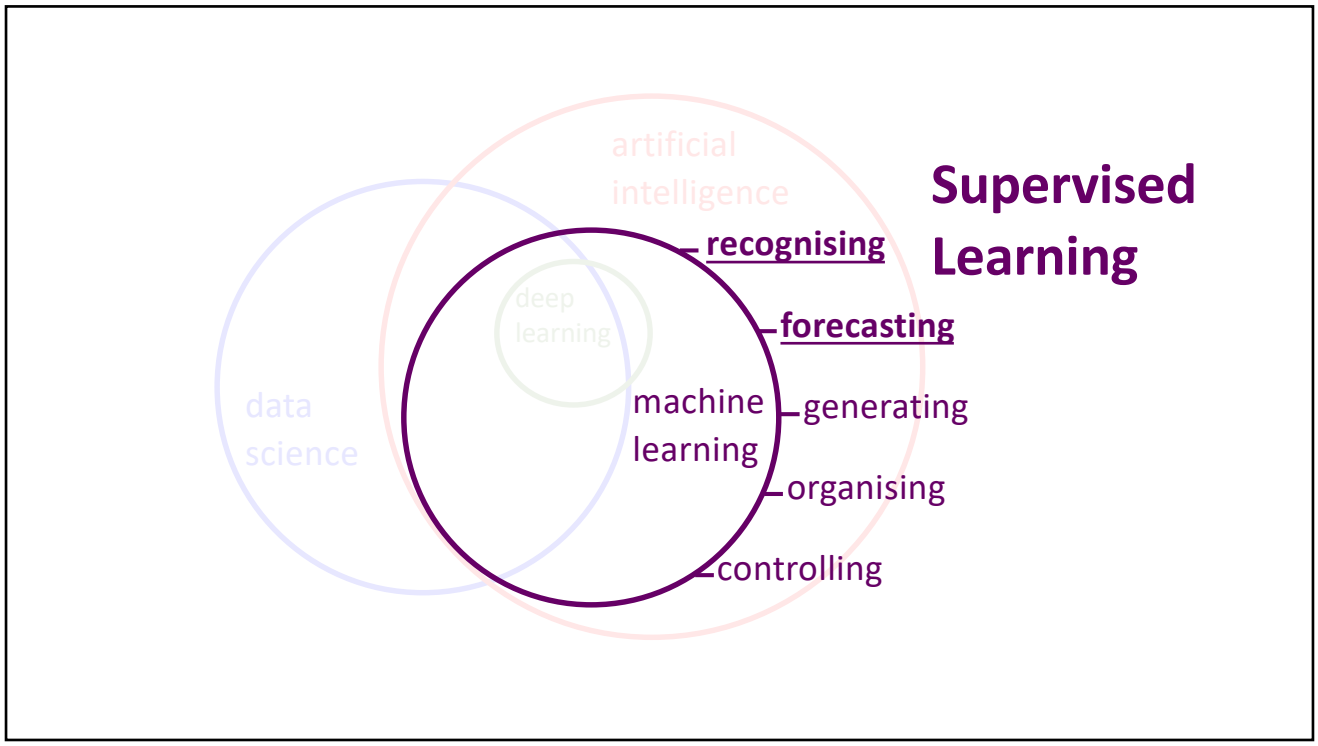


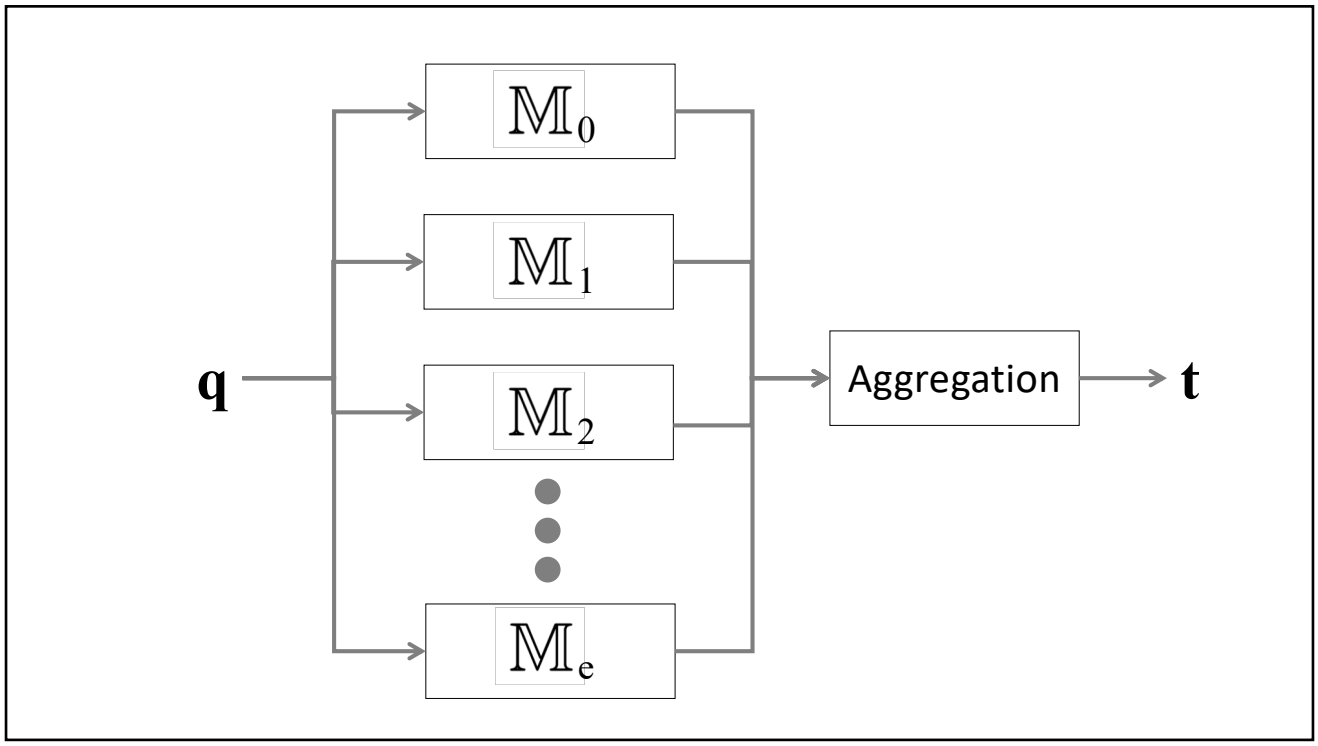
COMP47590
Advanced Machine Learning
Supervised Learning - Ensembles 2

Dr. Brian Mac Namee

Information

Email:	Brian.MacNamee@ucd.ie
Course Materials:	All material posted on UCD CS moodlehttps://csmoodle.ucd.ie/moodle/course/view.php?id=663 Enrolment key **UCDAdvML2017**


artificial intelligence
deep learning
machine learning
data science
recognising
forecasting
generating
organising
controlling
Supervised Learning


$\mathbb{M}_0$
$\mathbb{M}_1$
$\mathbb{M}_2$
$\mathbb{M}_e$
q
Aggregation
t

Practical Ensembles

There are however a series of pracical ensemble approaches

- Bagging
- Random forests
- Boosting
- Gradient boosting
- Stacking

Practical Ensembles

There are however a series of pracical ensemble approaches

- Bagging
- Random forests
- **Boosting**
- **Gradient boosting**
- **Stacking**

BOOSTING

Boosting

Boosting works by iteratively creating models and adding them to the ensemble

- Each new model added to the ensemble is biased to pay more attention to instances that previous models miss-classified
- This is done by incrementally adapting the dataset used to train the models
- The iteration stops when a predefined number of models have been added

Boosting

Boosting uses a weighted dataset

- Each instance has an associated weight $\mathbf{w}_i \geq 0$,
- Initially set to $1/n$ where n is the number of instances in the dataset
- After each model is added to the ensemble it is tested on the training data and the weights are adjusted
- These weights are used as a distribution over which the full dataset is sampled for each training dataset

Boosting

During each training iteration the algorithm:

- Induces a model and calculates the total error, ε, by summing the weights of the training instances for which the predictions made by the model are incorrect.

Boosting

During each training iteration the algorithm:

- Increases the weights for the instances misclassified using:

$$\mathbf{w}[i] \leftarrow \mathbf{w}[i] \times \left(\frac{1}{2 \times \epsilon}\right)$$

- Decreases the weights for the instances correctly classified:

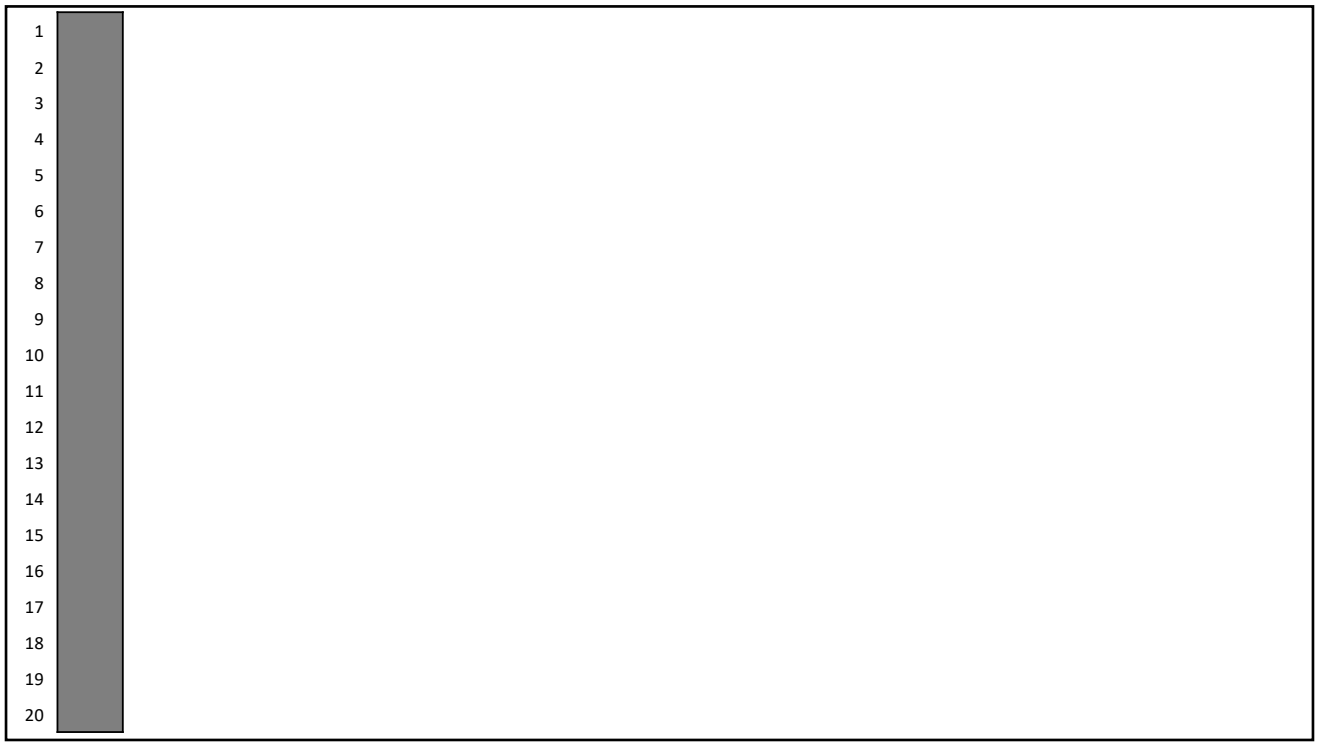
$$\mathbf{w}[i] \leftarrow \mathbf{w}[i] \times \left(\frac{1}{2 \times (1 - \epsilon)}\right)$$

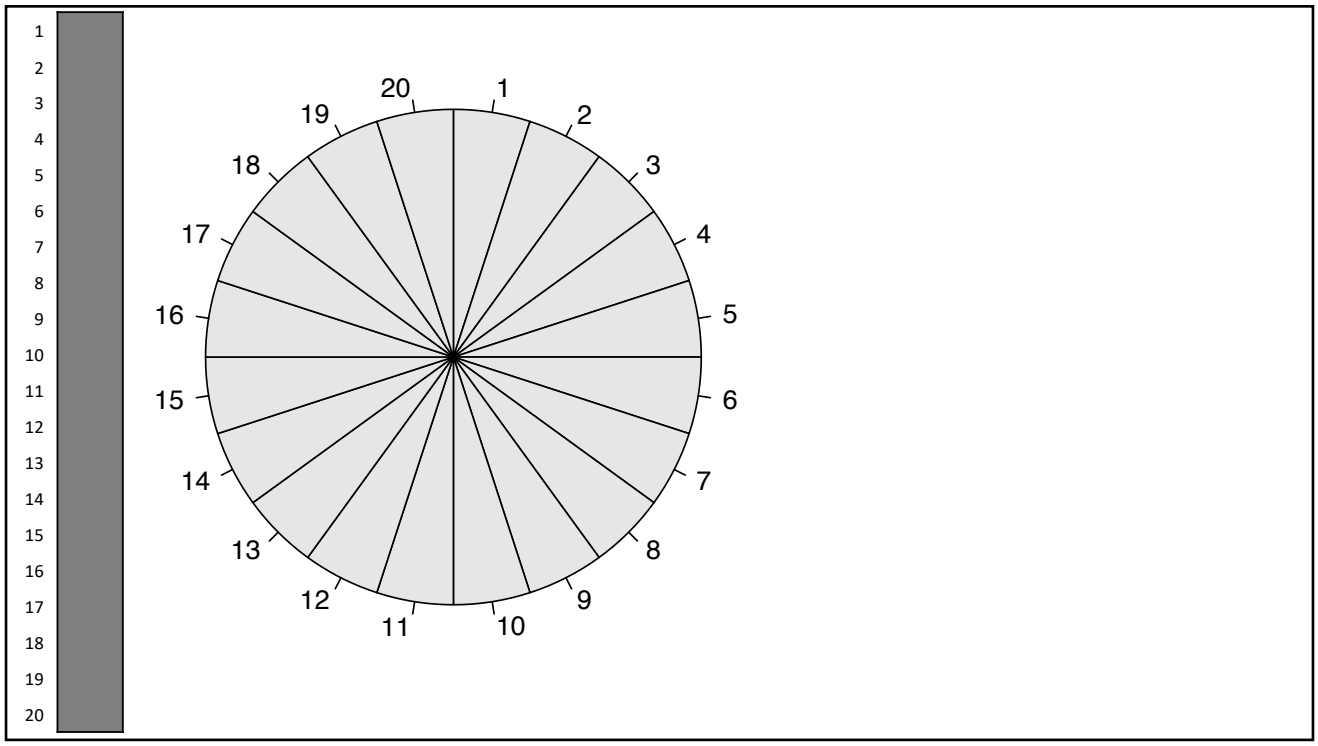
Boosting

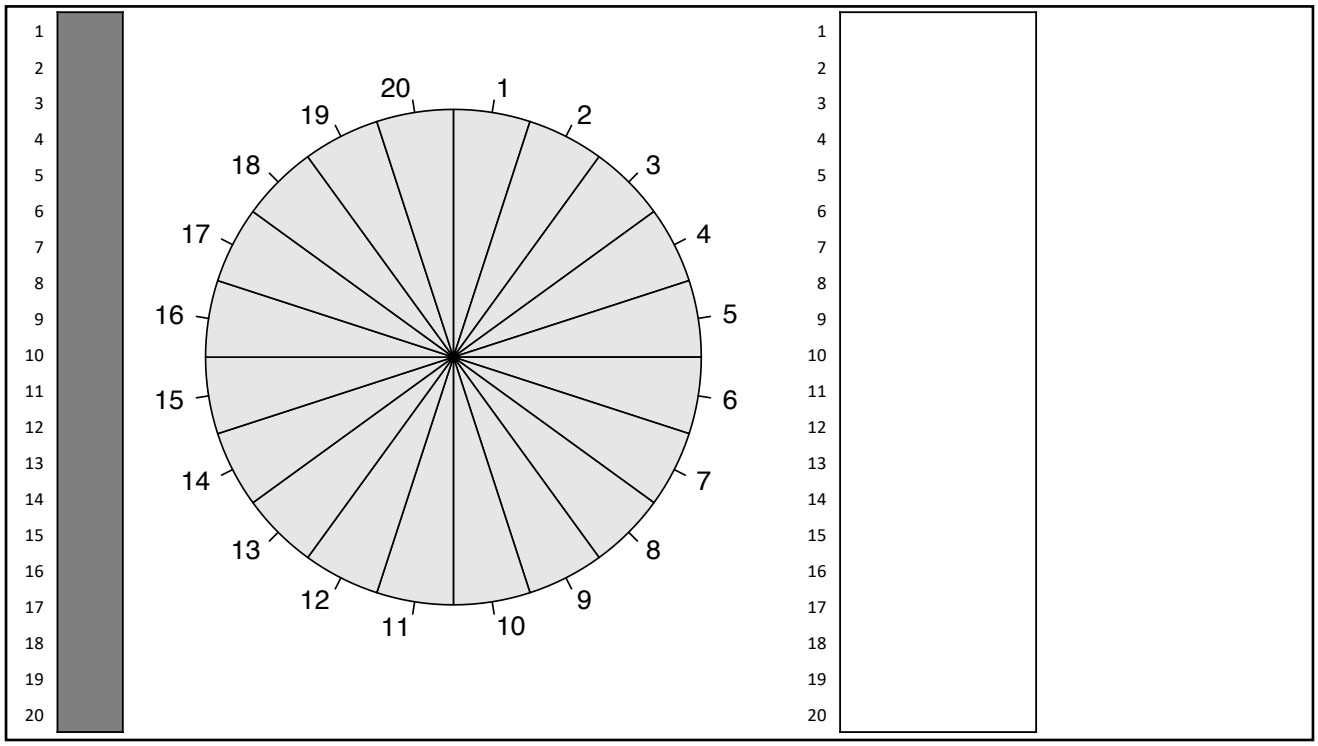
During each training iteration the algorithm:

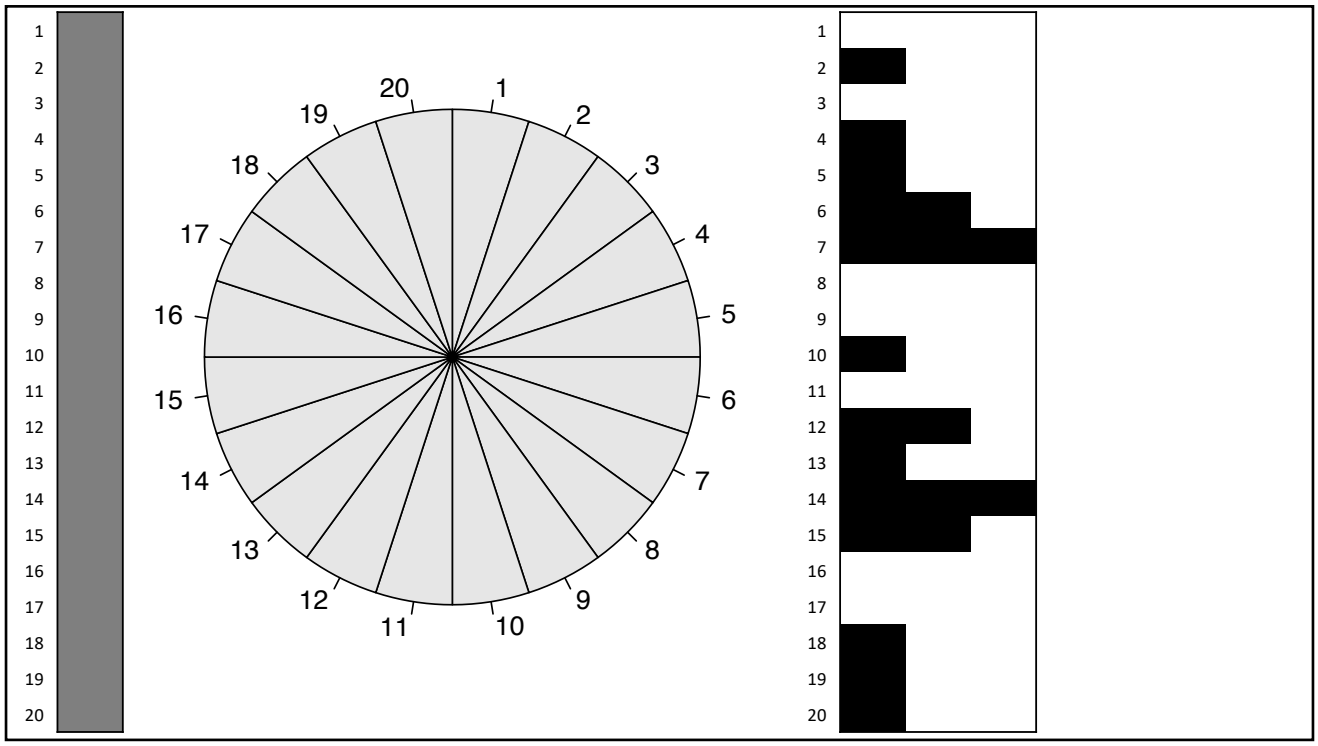
- Calculate a confidence factor, α, for the model such that α increases as ε decreases:

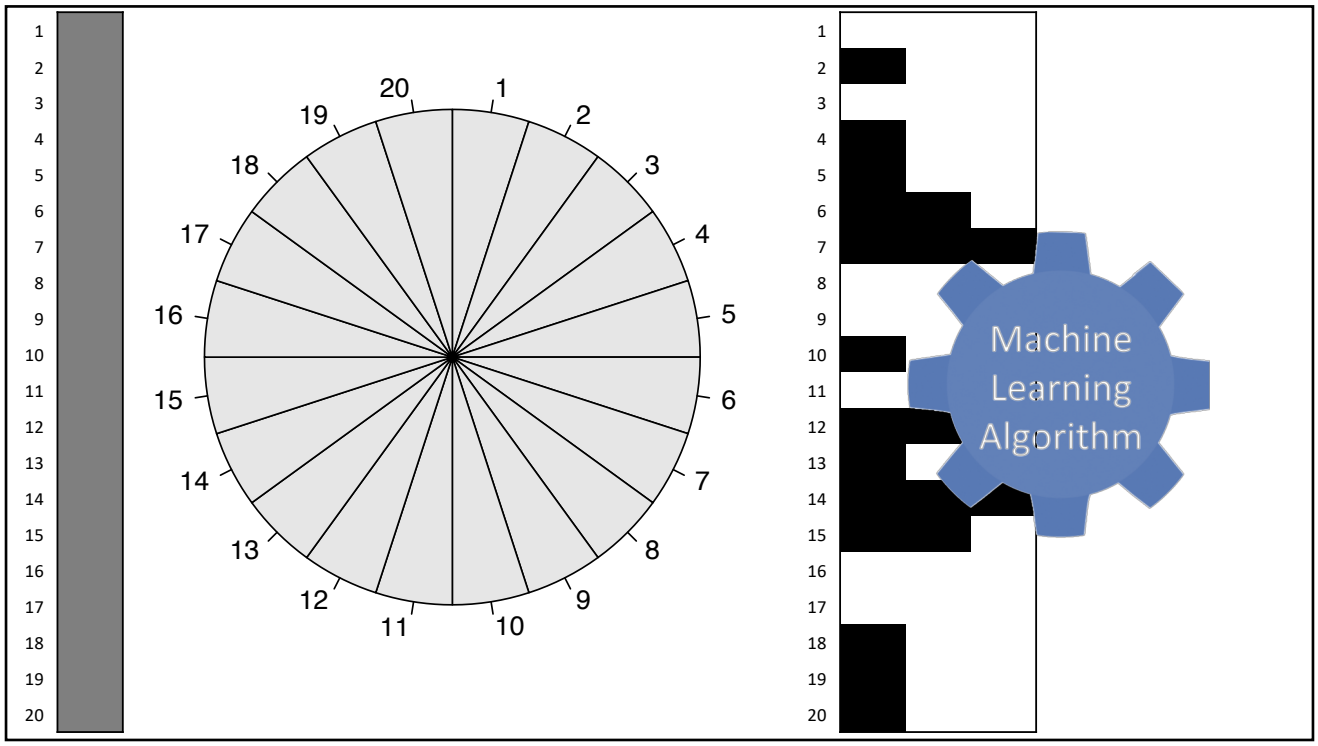
$$\alpha = \frac{1}{2} \times log_e\left(\frac{1 - \epsilon}{\epsilon}\right)$$

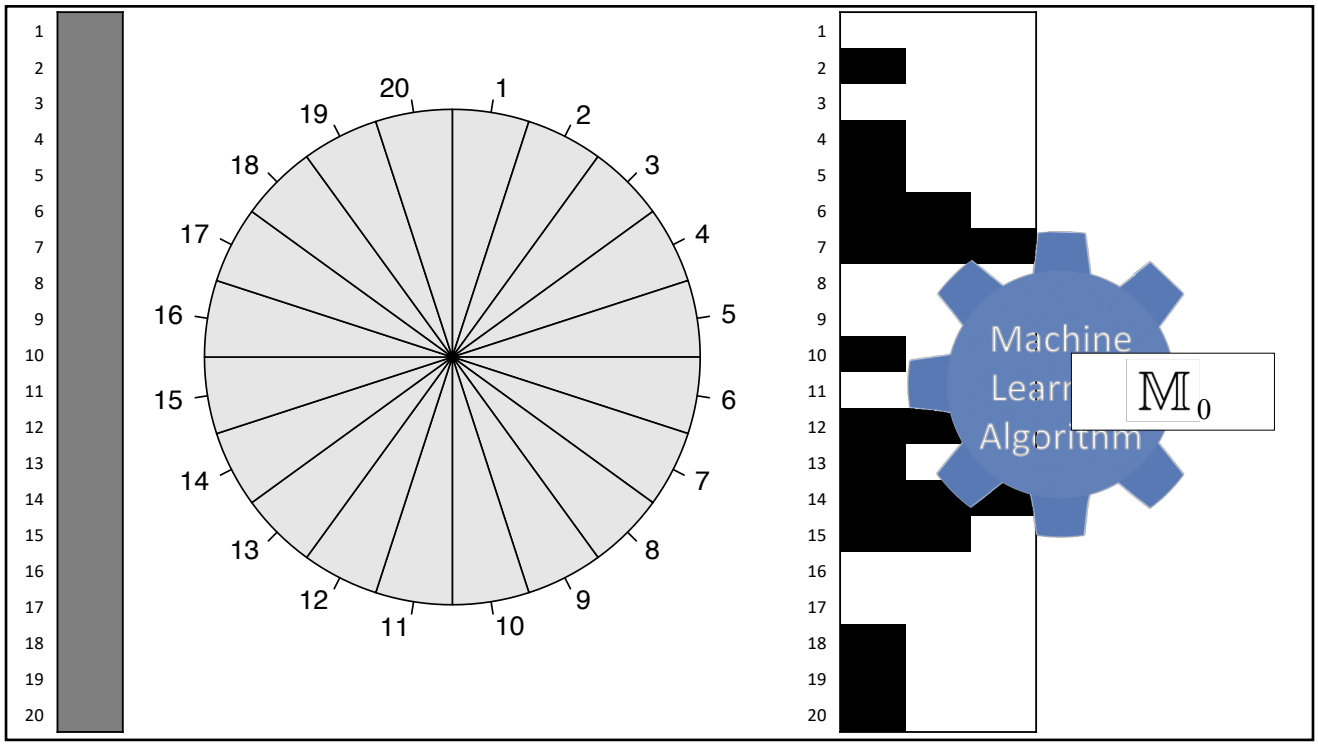

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20

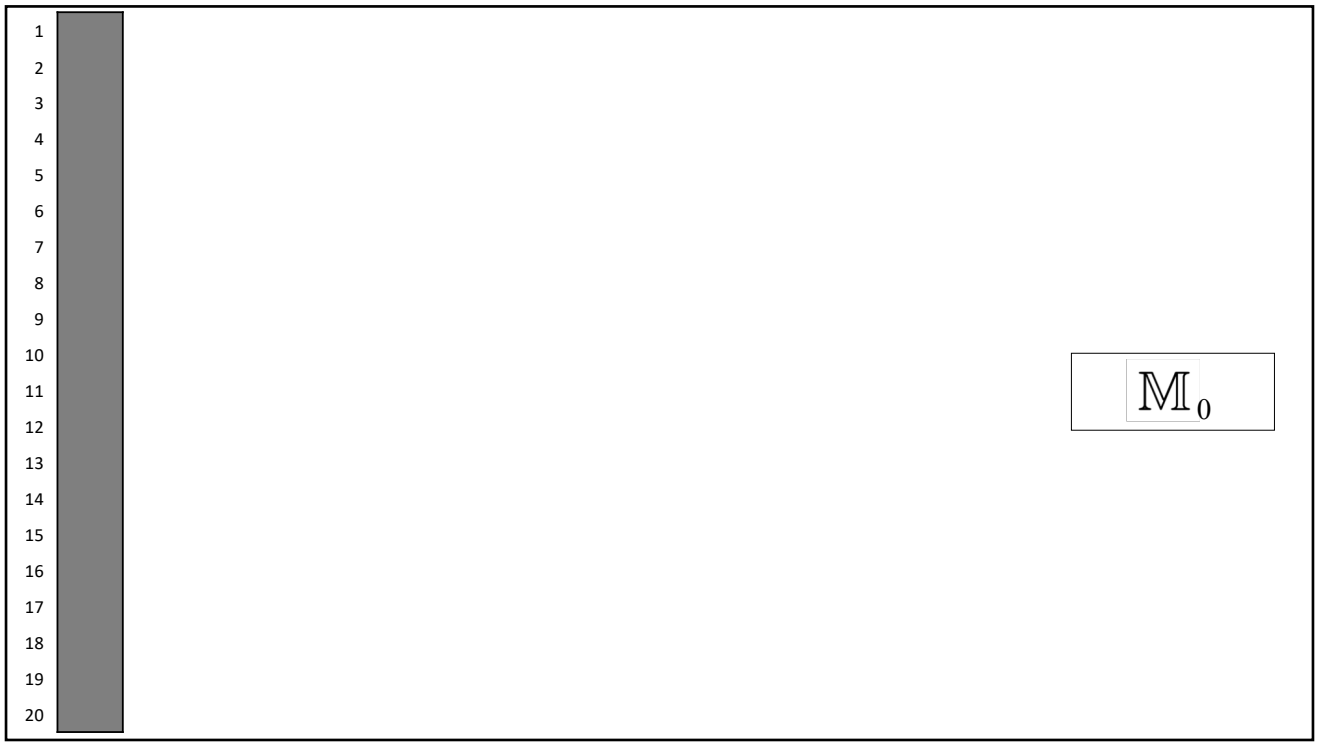

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20

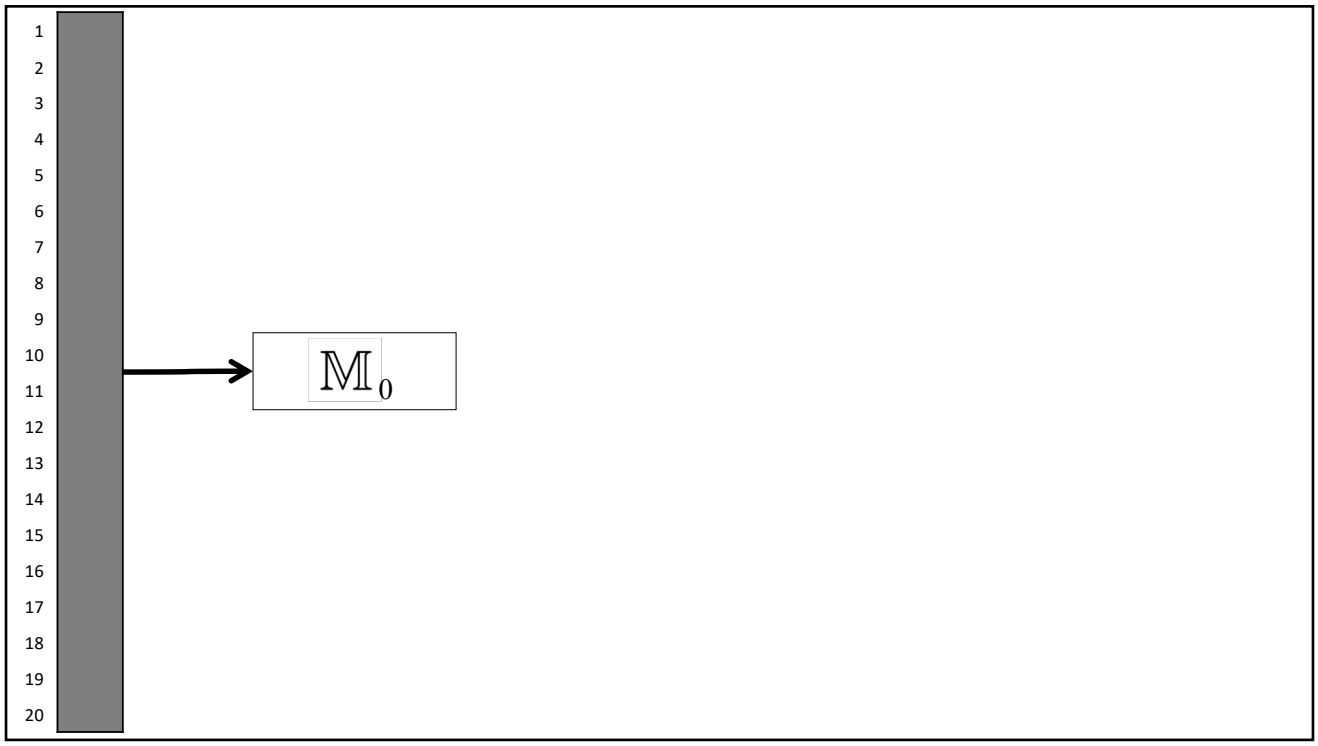

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
20 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

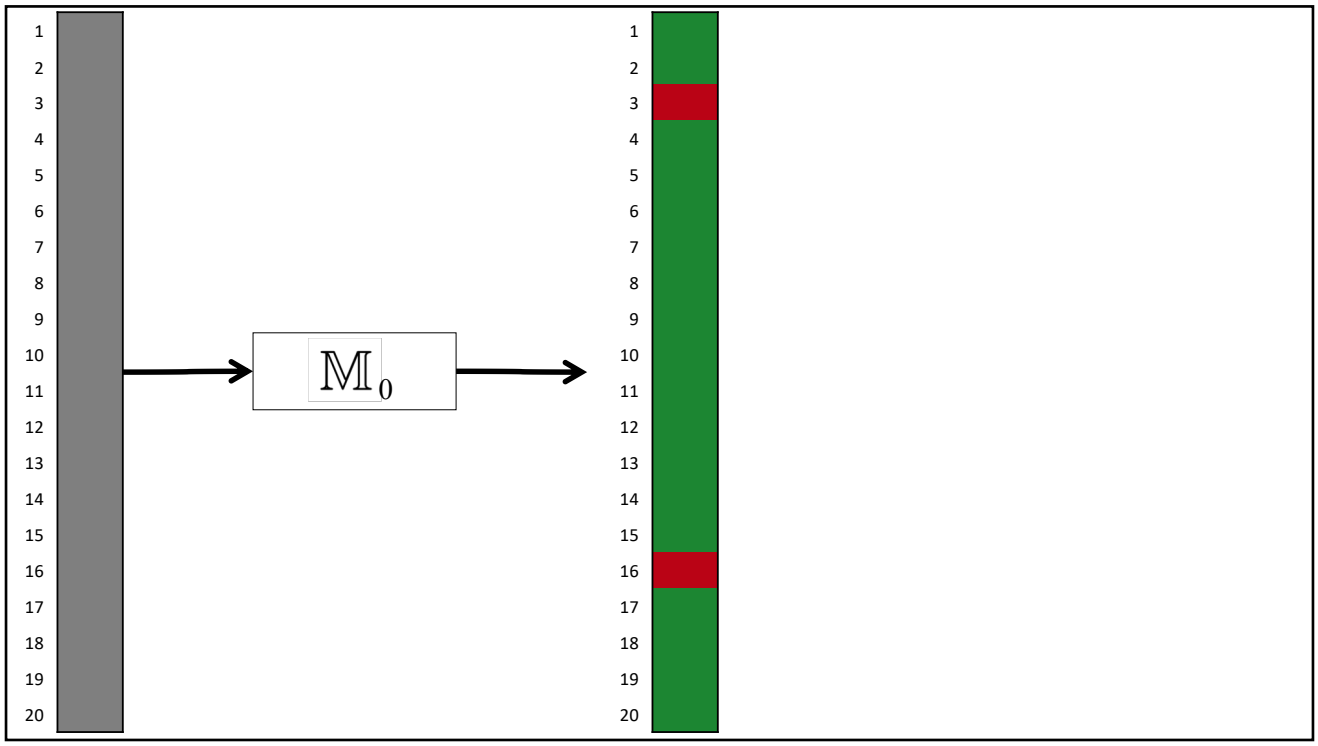

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
20 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

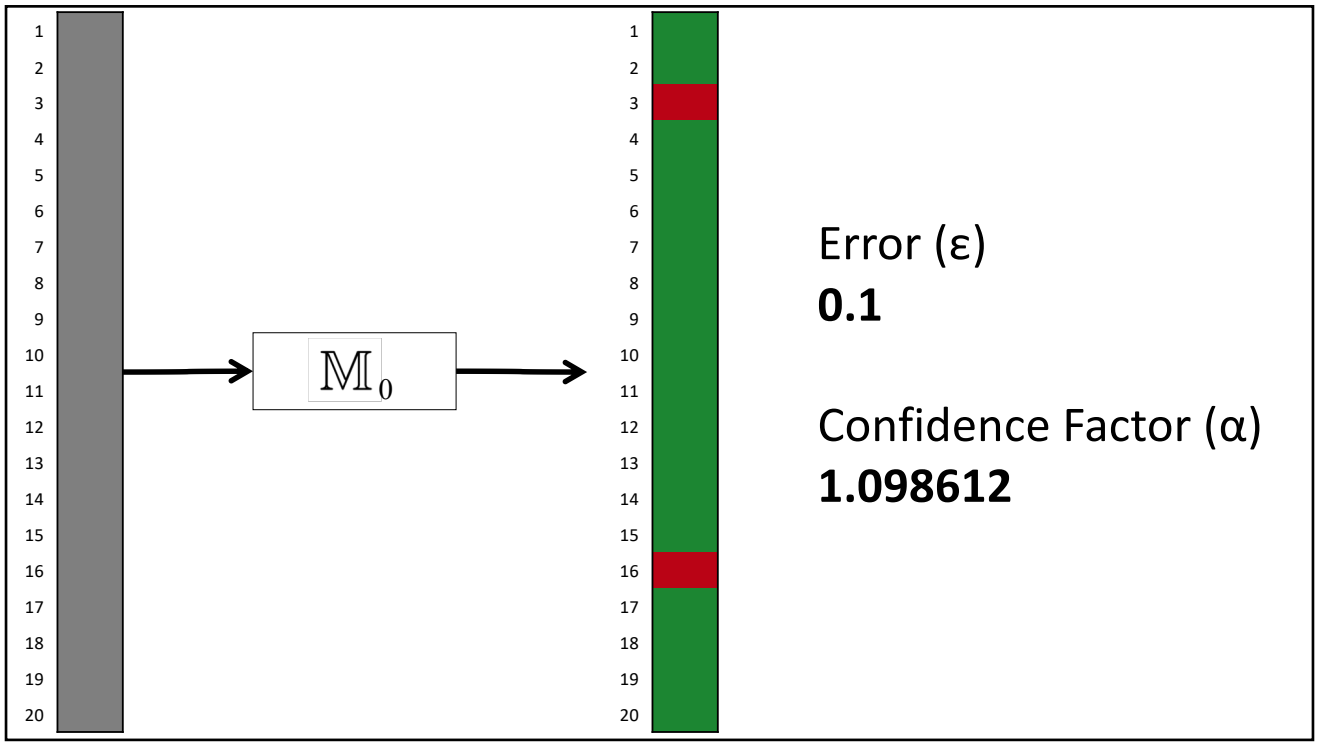

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
Machine
Learning
Algorithm

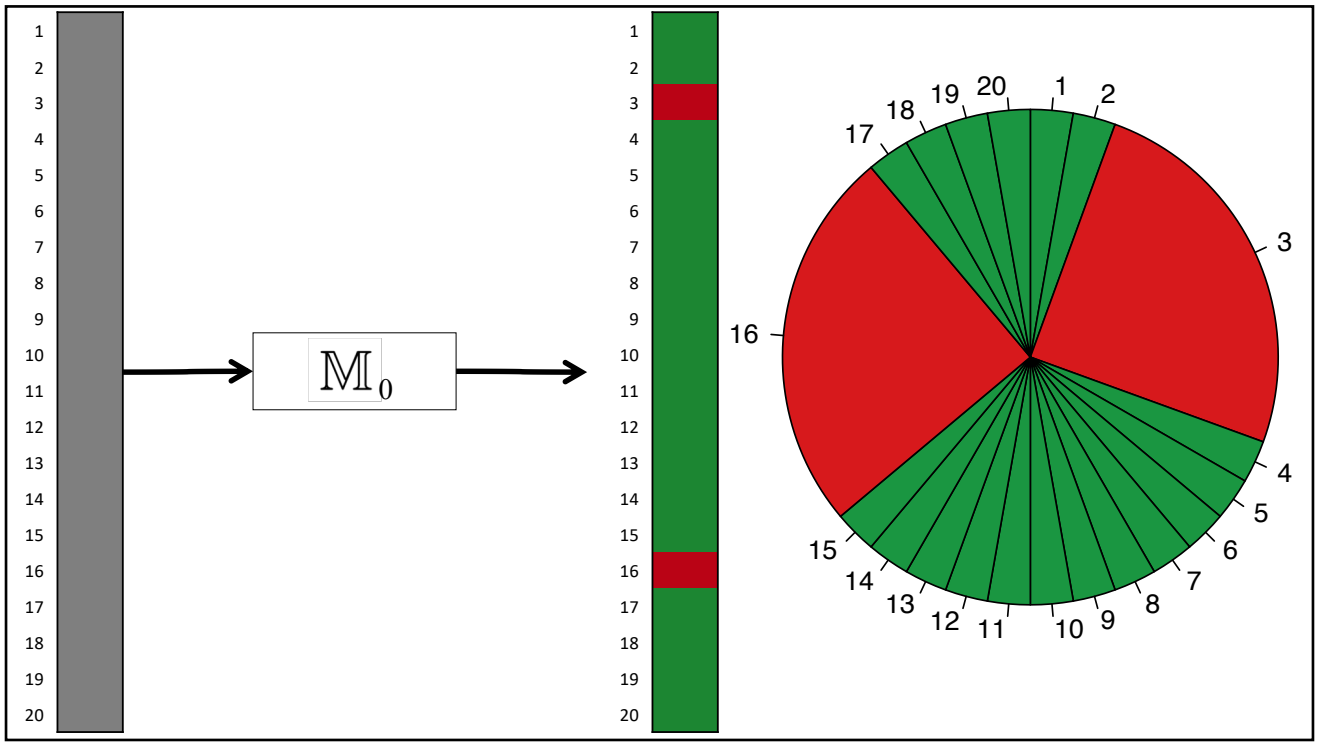

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
Machine
Learn
Algorithm
$\mathbb{M}_0$

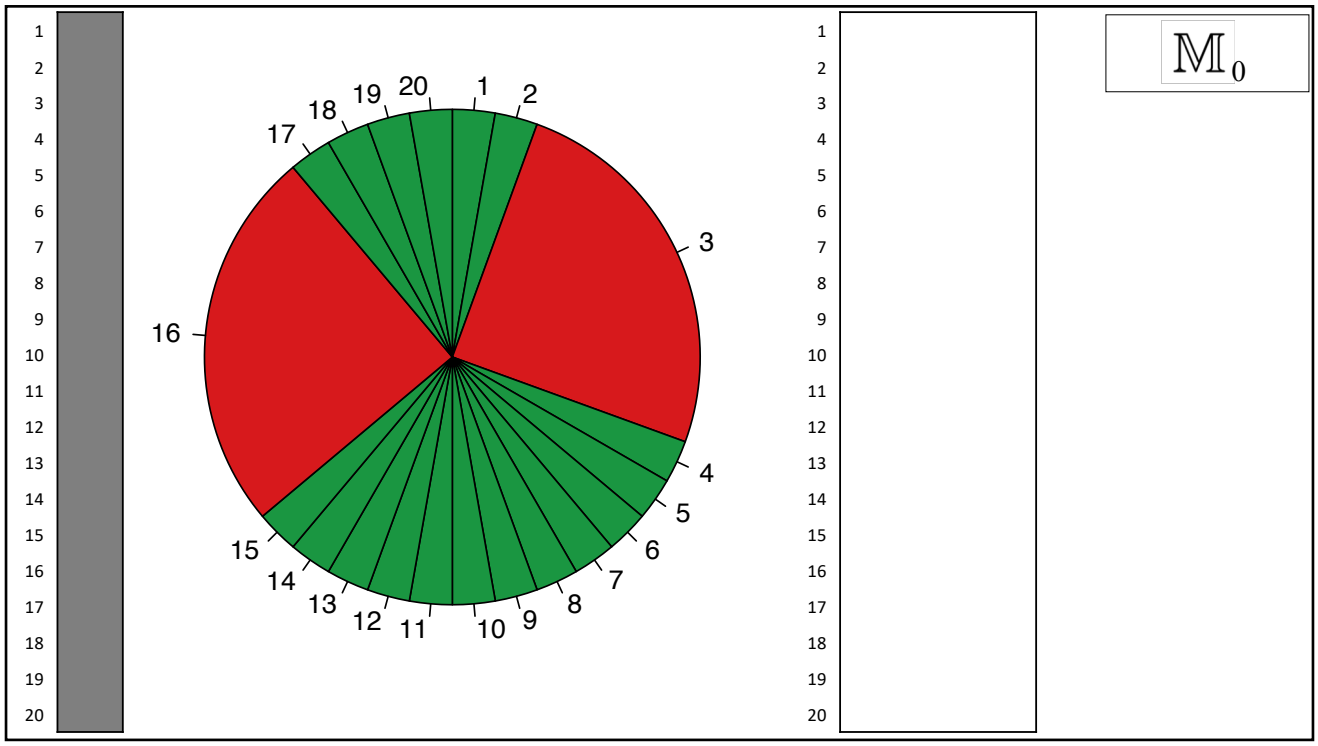

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
$\mathbb{M}_0$

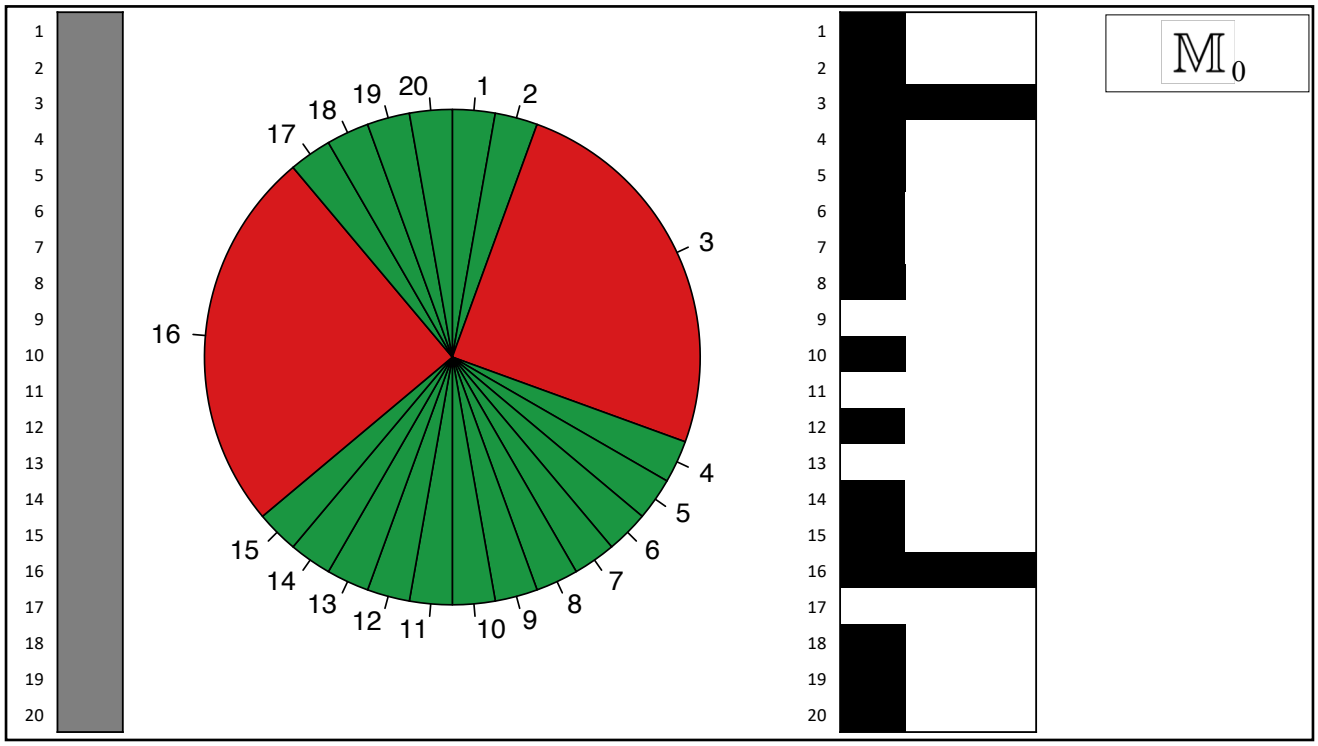

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
$\mathbb{M}_0$

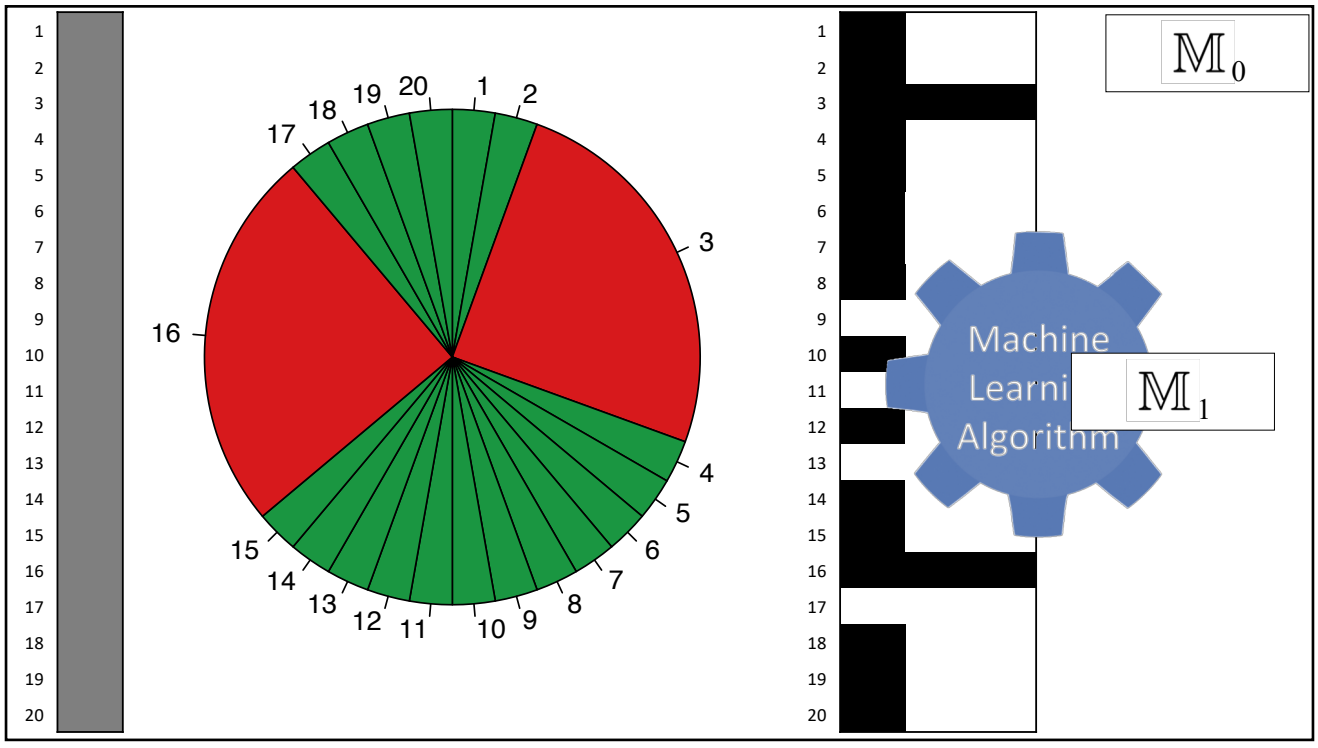

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
$\mathbb{M}_0$

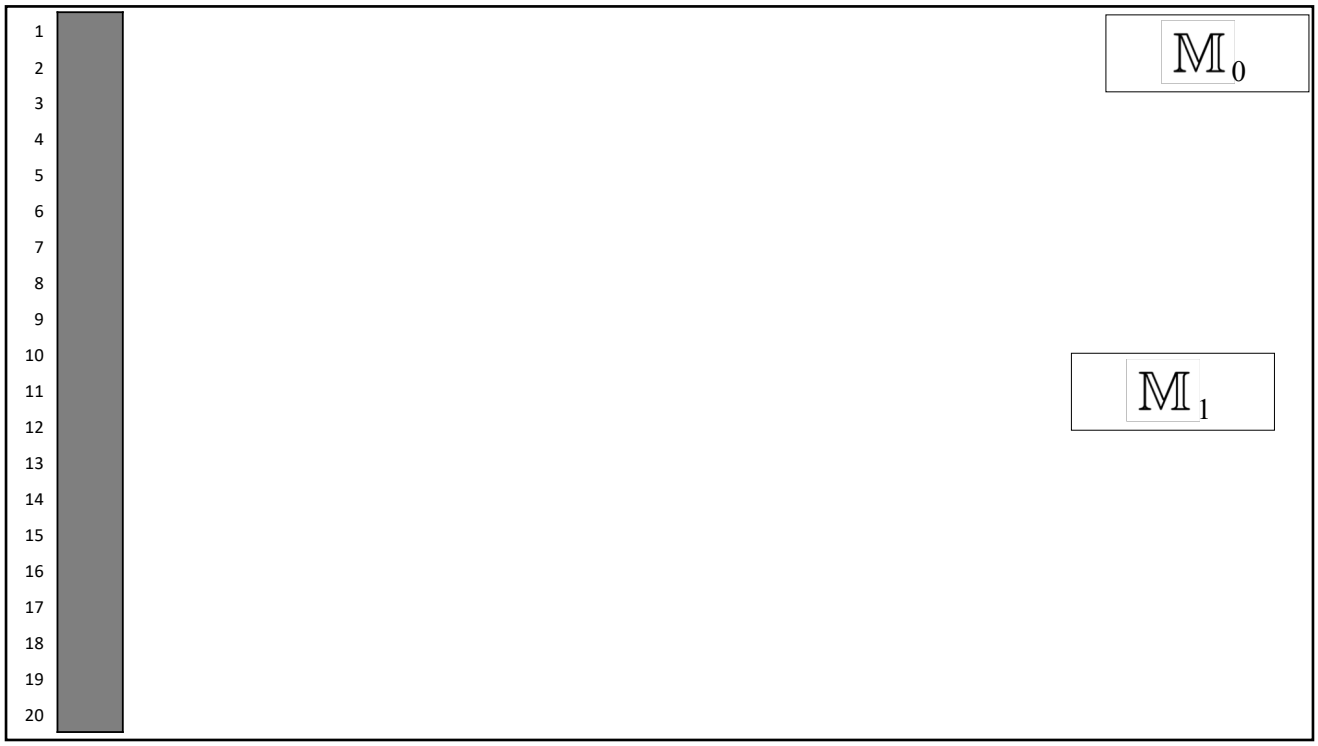

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
$\mathbb{M}_0$
Error (ε)
0.1
Confidence Factor (α)
1.098612

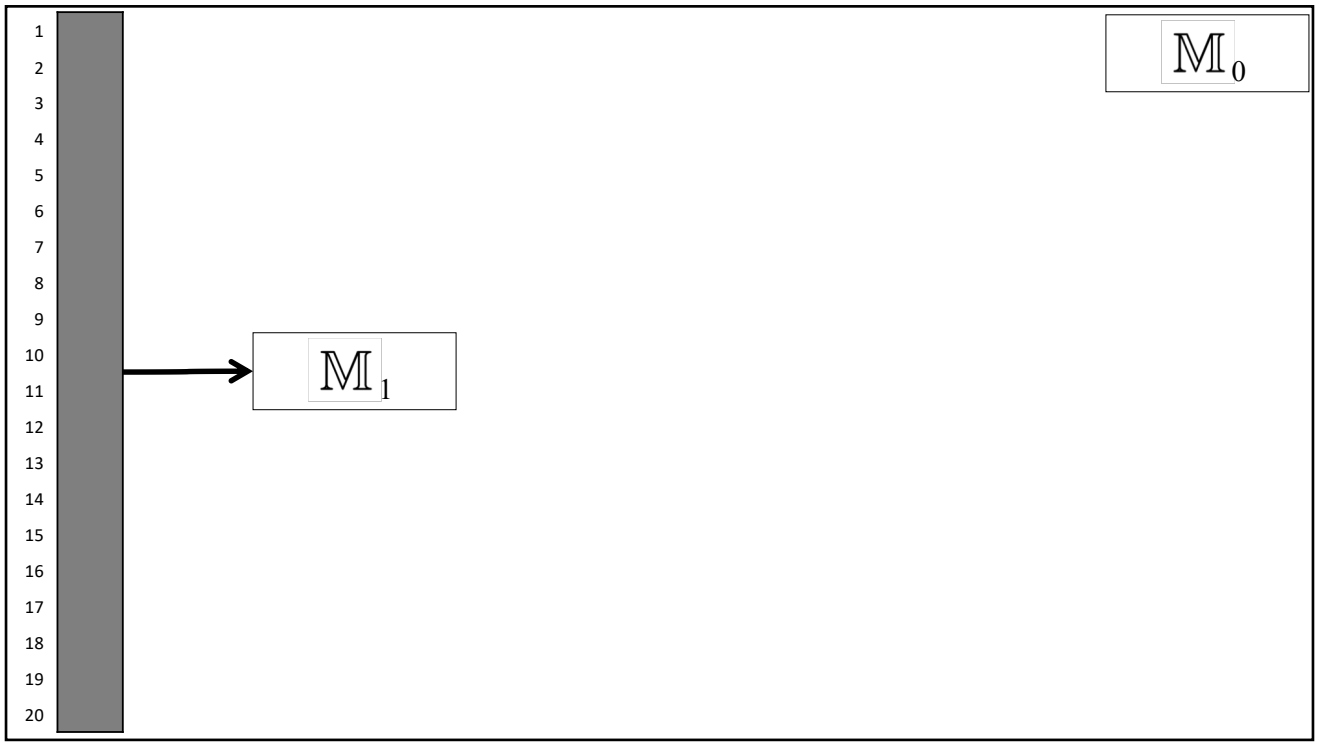

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
$\mathbb{M}_0$
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

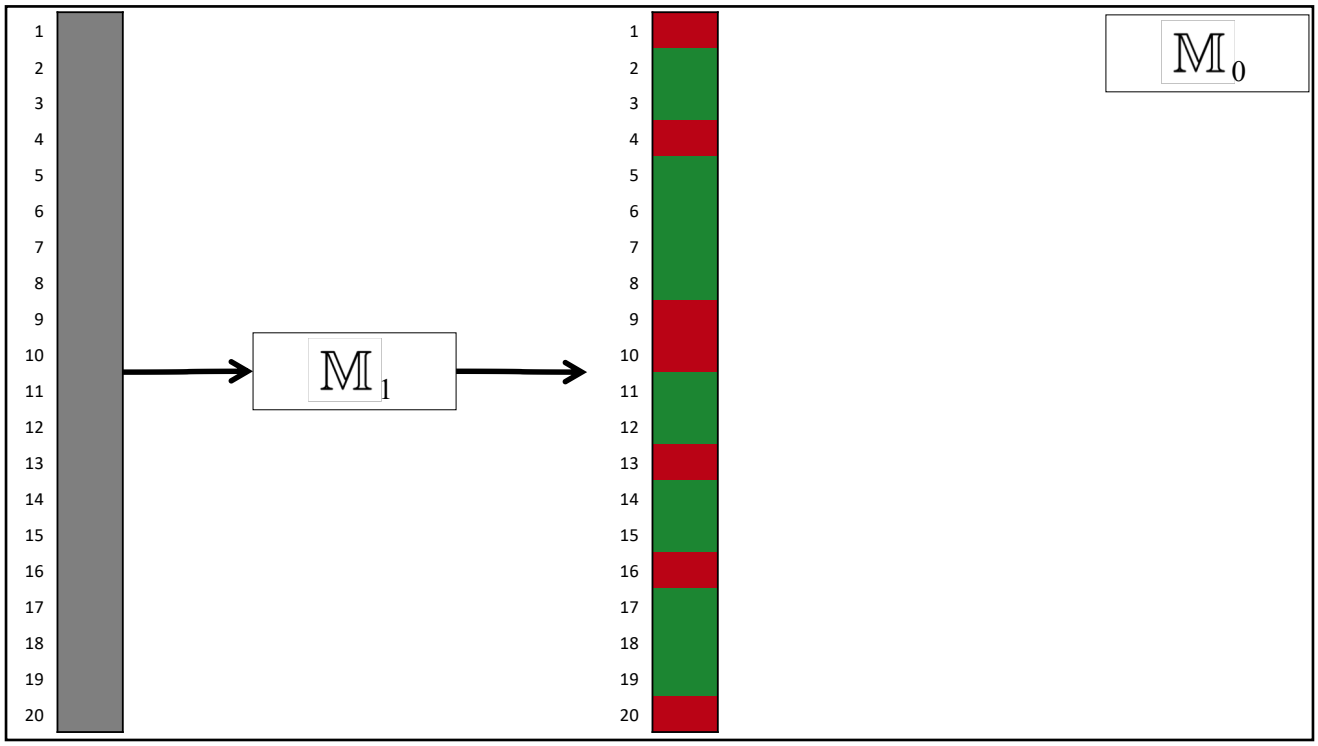

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
$\mathbb{M}_0$

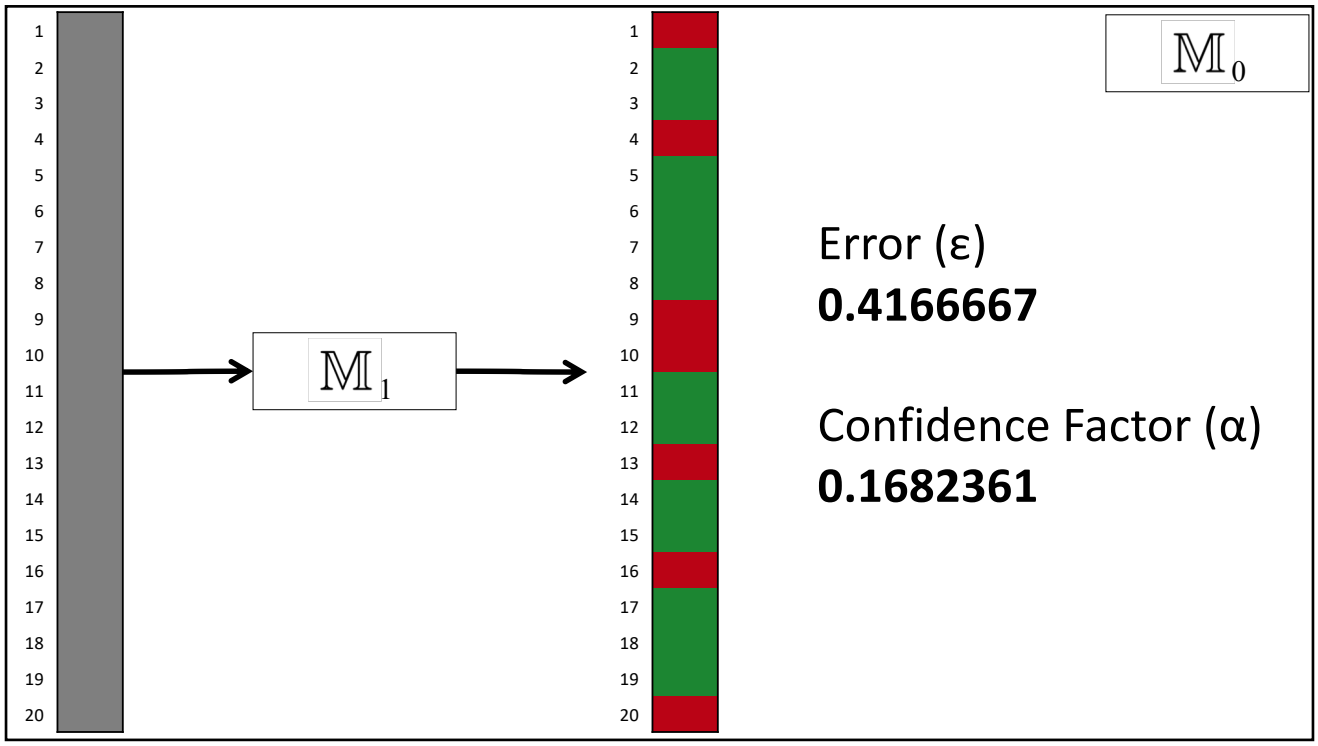

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
$\mathbb{M}_0$

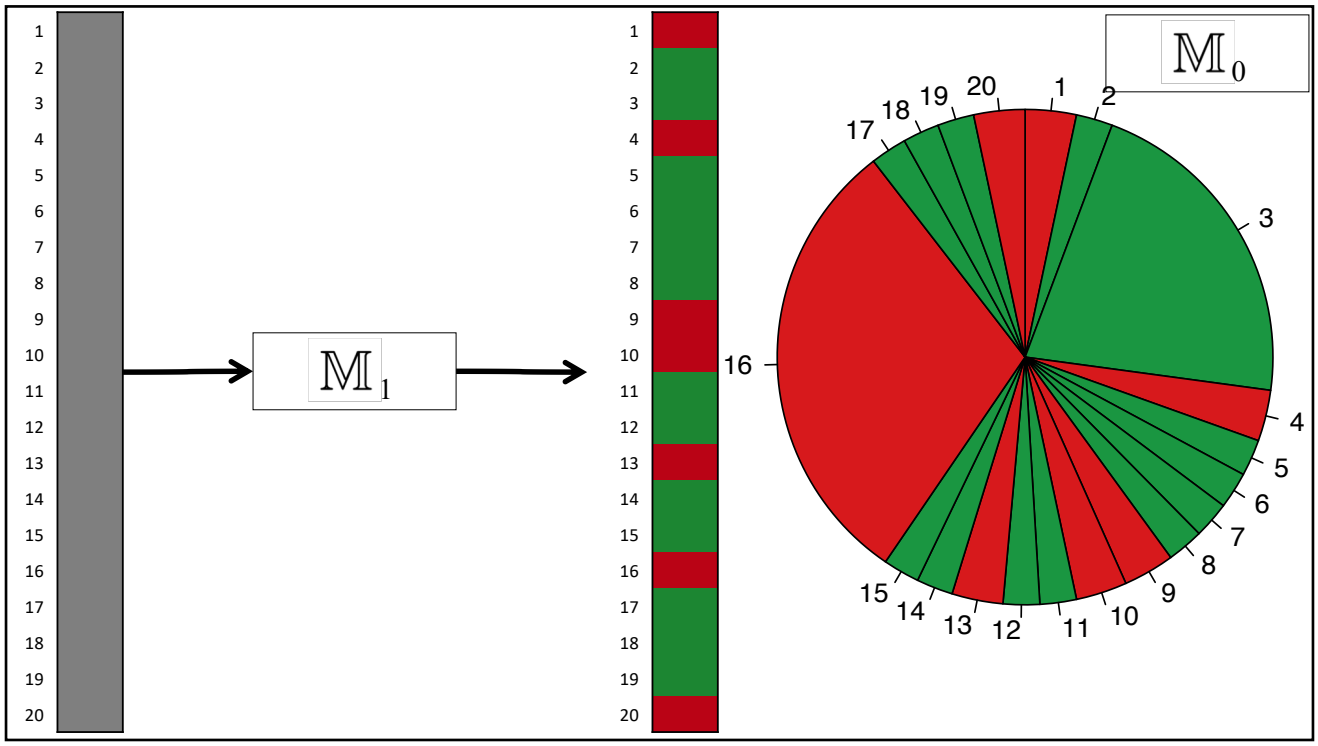

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
$\mathbb{M}_0$
Machine Learni Algorithm
$\mathbb{M}_1$

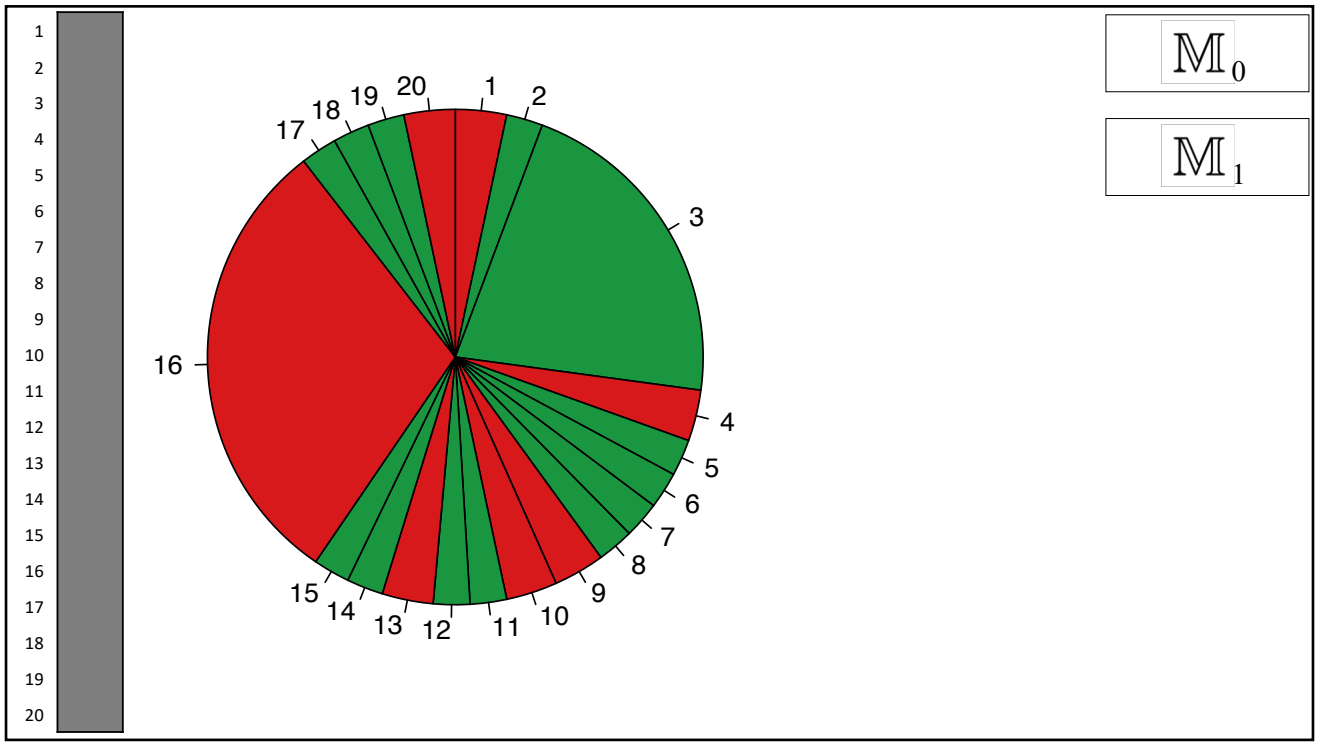

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
$\mathbb{M}_0$
$\mathbb{M}_1$

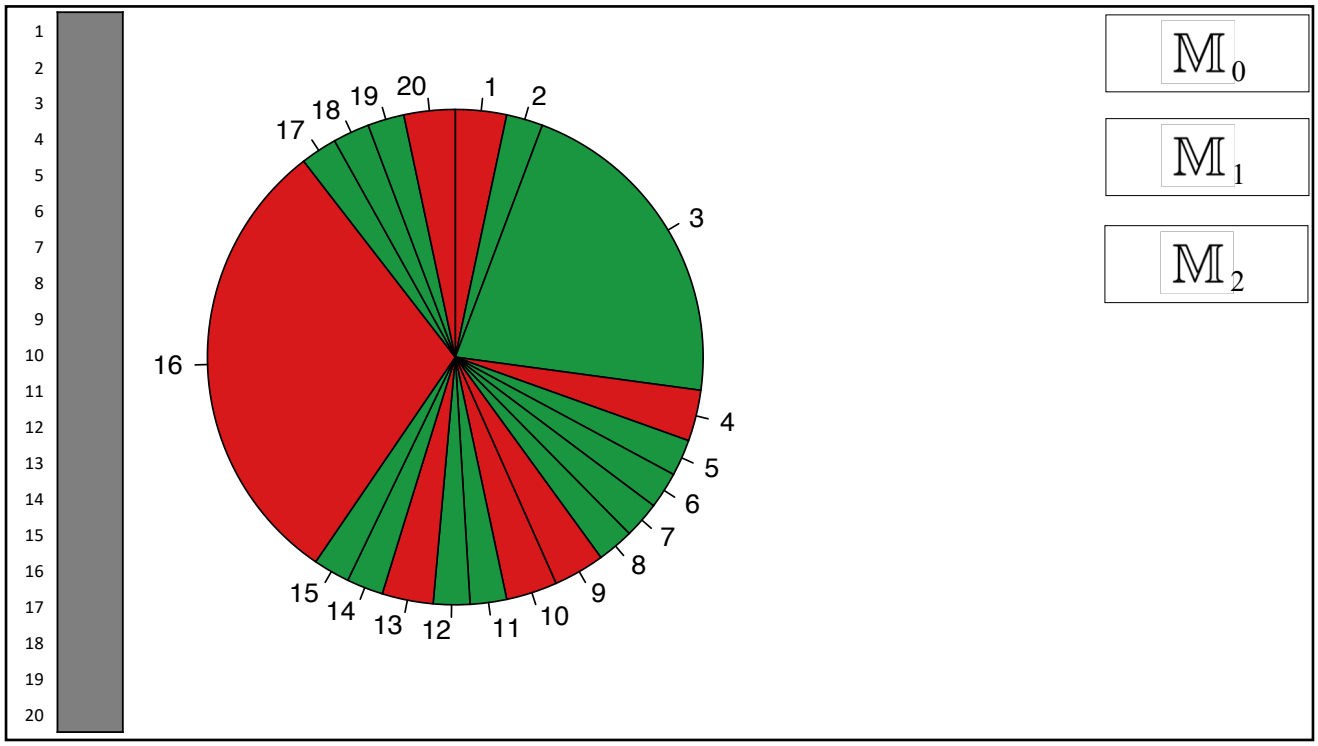

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
$\mathbb{M}_0$
$\mathbb{M}_1$

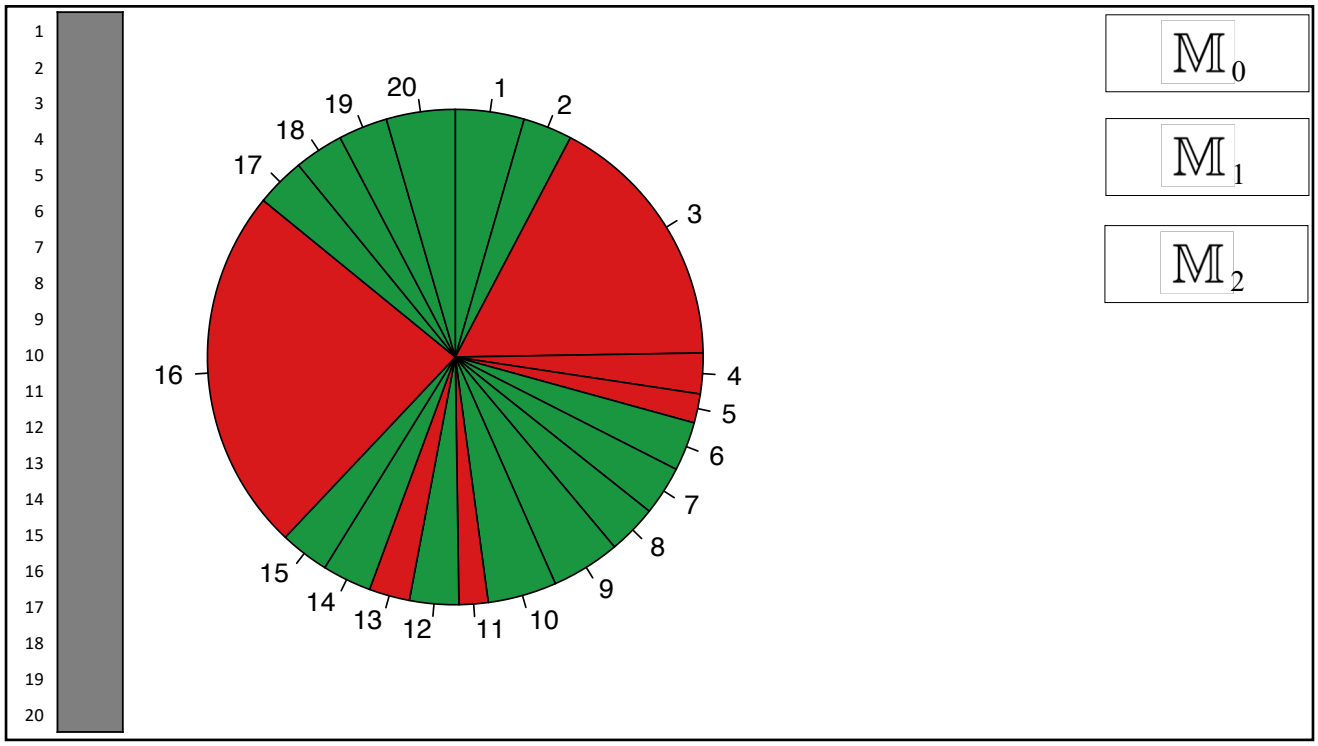

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
$\mathbb{M}_1$
$\mathbb{M}_0$

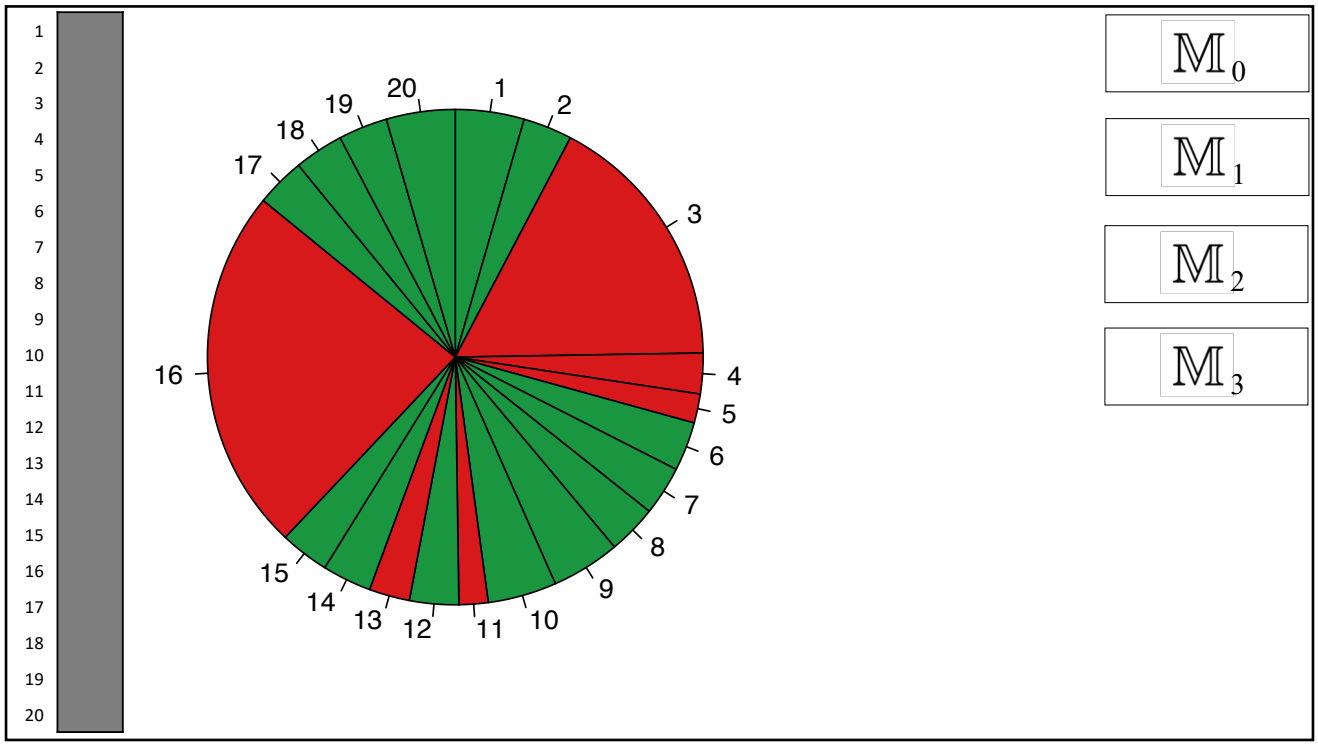

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
$\mathbb{M}_1$
$\mathbb{M}_0$
Error (ε)
0.4166667
Confidence Factor (α)
0.1682361

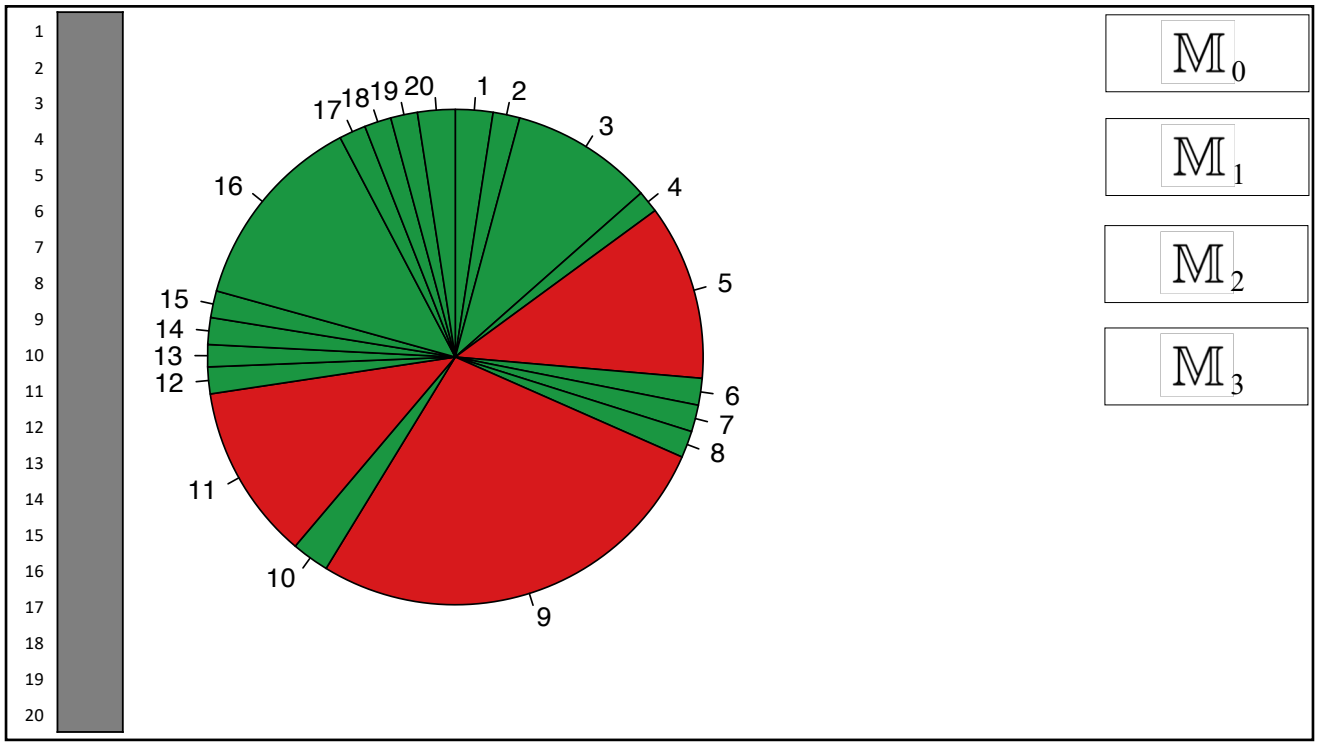

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
$\mathbb{M}_1$
$\mathbb{M}_0$
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20

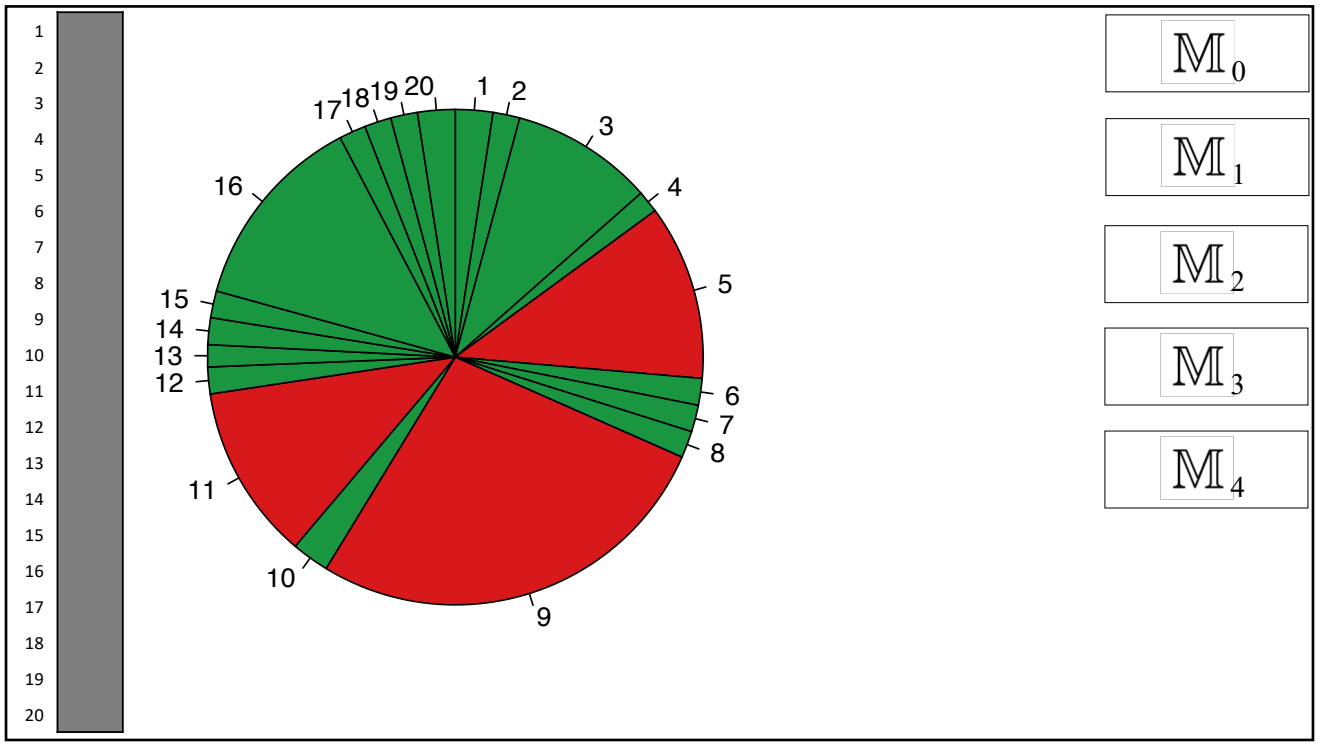

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
$\mathbb{M}_0$
$\mathbb{M}_1$
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20

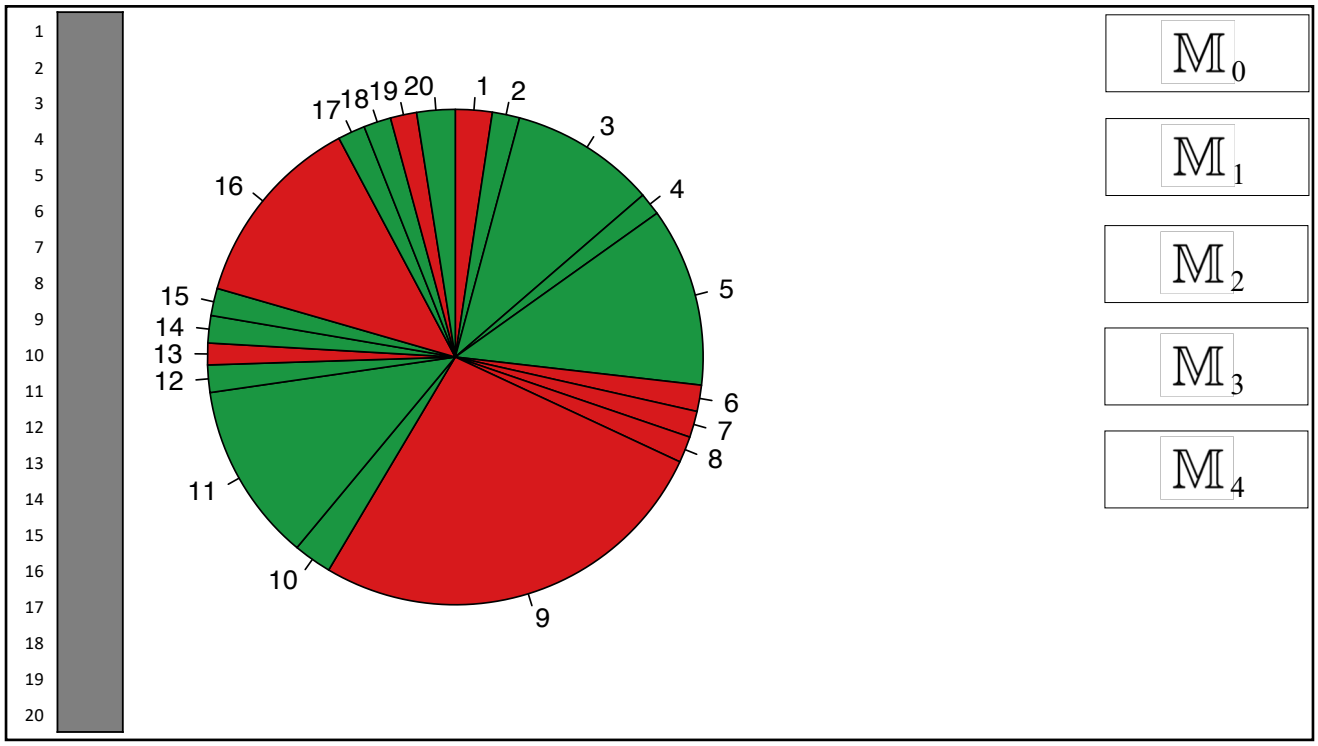

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
$\mathbb{M}_0$
$\mathbb{M}_1$
$\mathbb{M}_2$

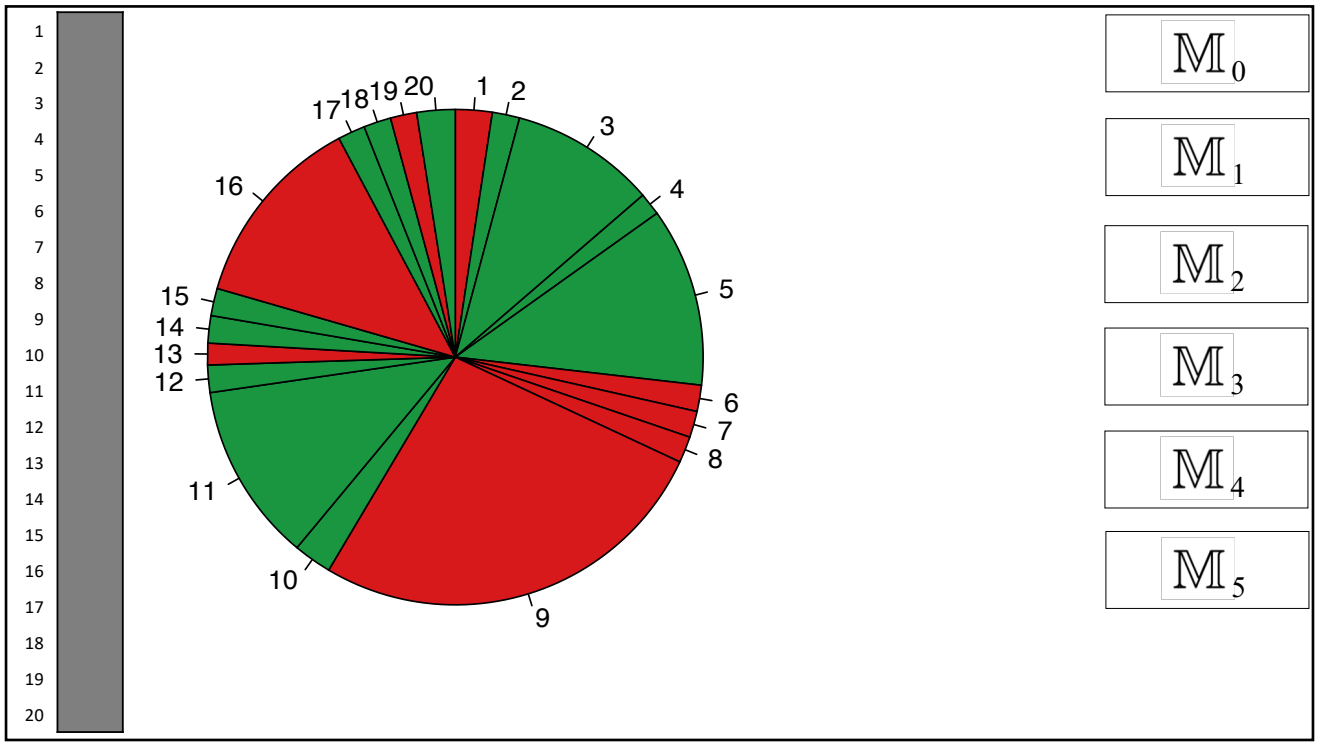

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
$\mathbb{M}_0$
$\mathbb{M}_1$
$\mathbb{M}_2$


1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
$\mathbb{M}_0$
$\mathbb{M}_1$
$\mathbb{M}_2$
$\mathbb{M}_3$


1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
$\mathbb{M}_0$
$\mathbb{M}_1$
$\mathbb{M}_2$
$\mathbb{M}_3$


1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
$\mathbb{M}_0$
$\mathbb{M}_1$
$\mathbb{M}_2$
$\mathbb{M}_3$
$\mathbb{M}_4$


1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
$\mathbb{M}_0$
$\mathbb{M}_1$
$\mathbb{M}_2$
$\mathbb{M}_3$
$\mathbb{M}_4$



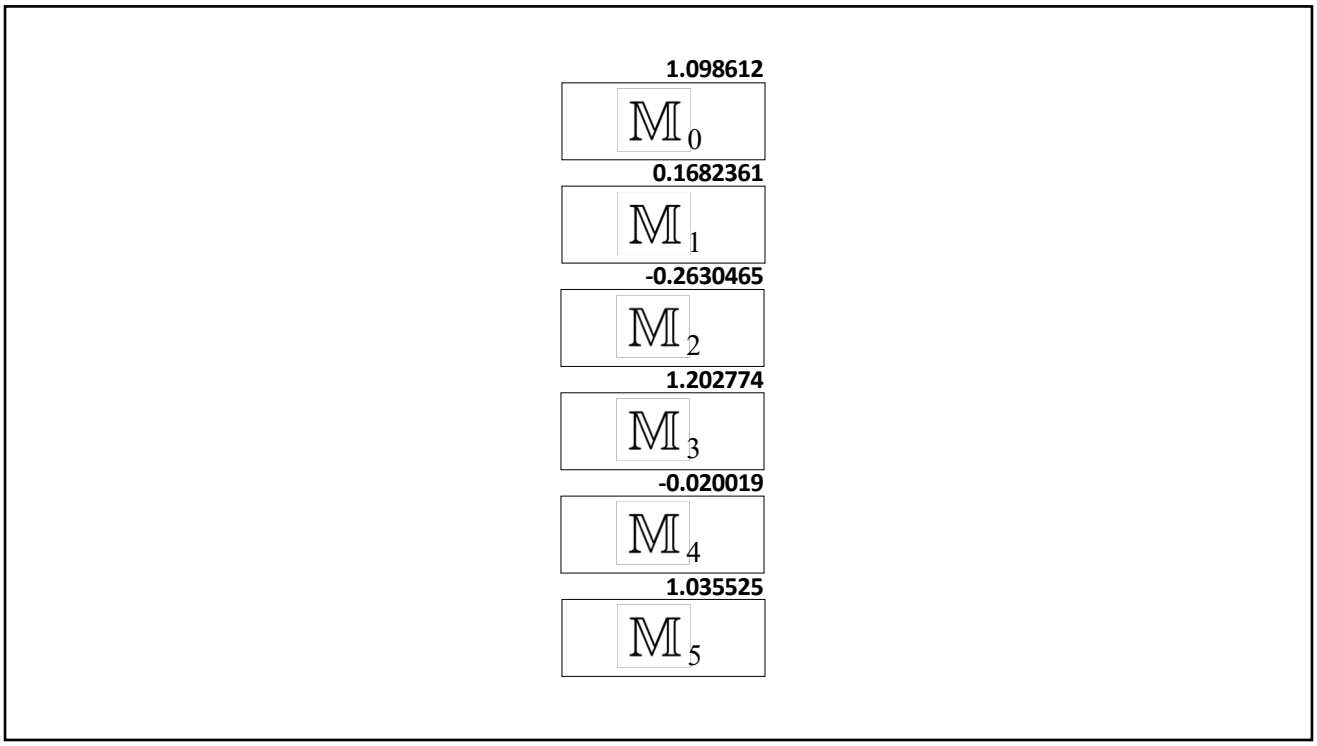
Boosting

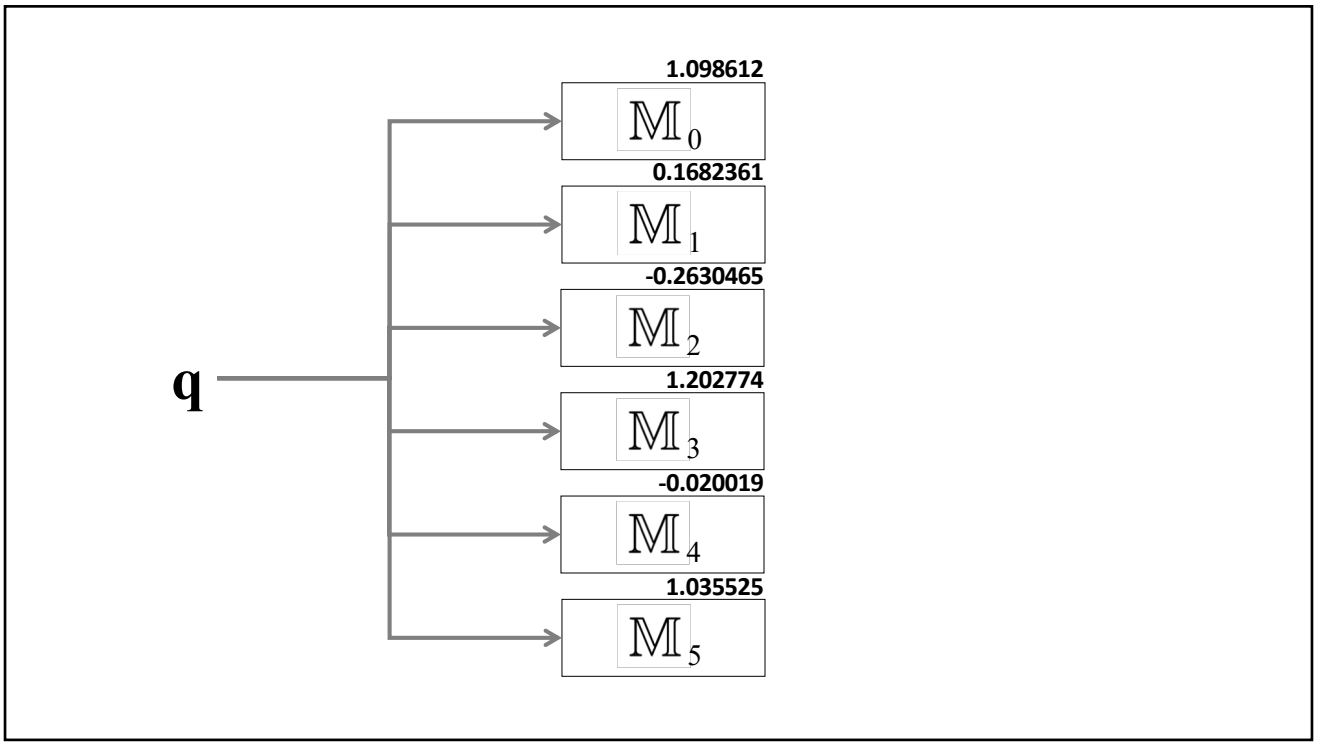
Predictions are made using a weighted aggregate of the individual models

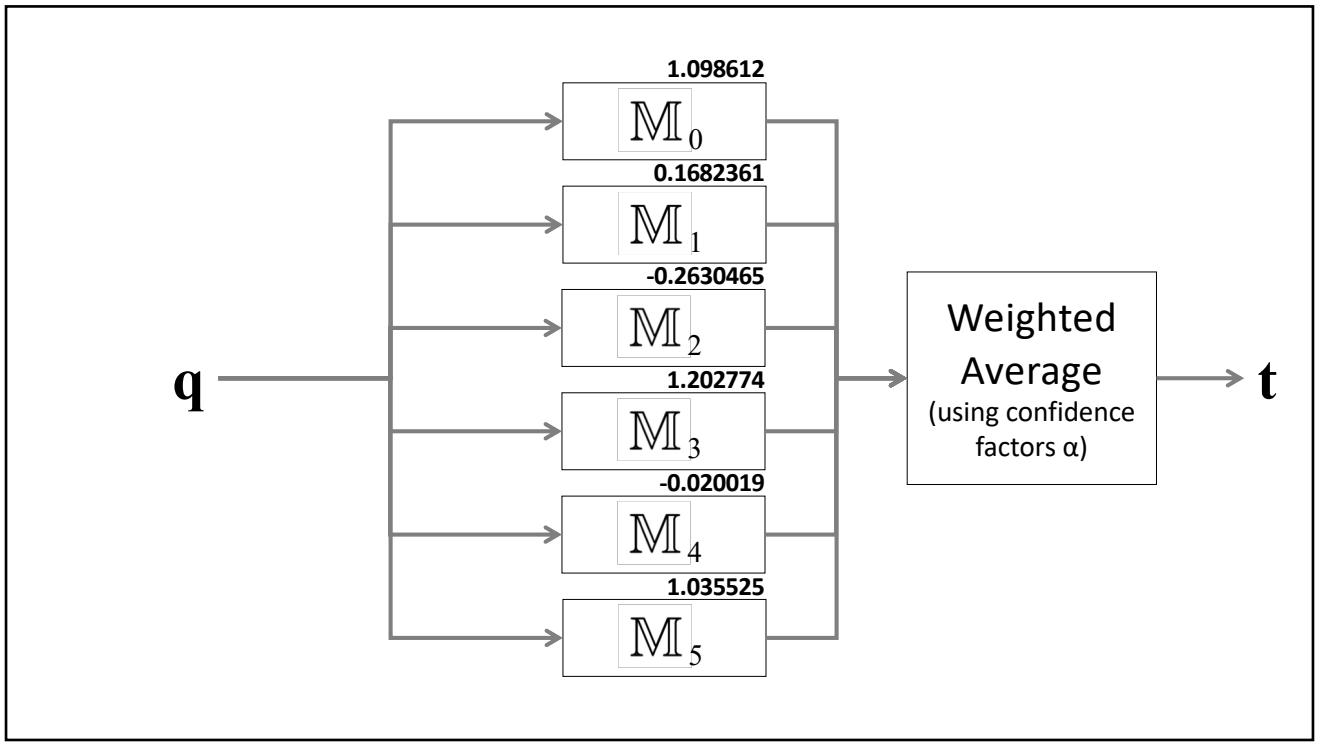
- Weights are based on confidence factors

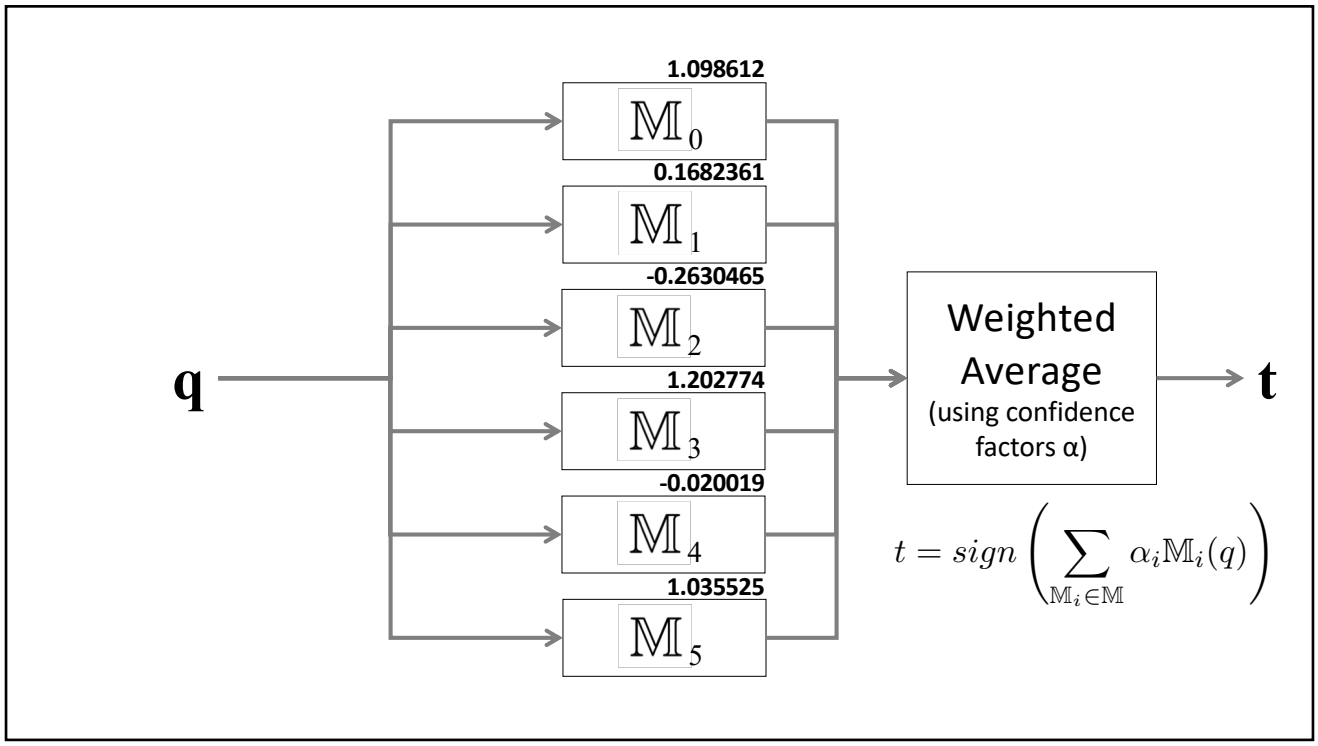
$$t = sign\left(\sum_{\mathbb{M}_i \in \mathbb{M}} \alpha_i \mathbb{M}_i(q)\right)$$

- Assumes binary outputs of +1 or -1


1.098612
$\mathbb{M}_0$
0.1682361
$\mathbb{M}_1$
-0.2630465
$\mathbb{M}_2$
1.202774
$\mathbb{M}_3$
-0.020019
$\mathbb{M}_4$
1.035525
$\mathbb{M}_5$


q
1.098612
$\mathbb{M}_0$
0.1682361
$\mathbb{M}_1$
-0.2630465
$\mathbb{M}_2$
1.202774
$\mathbb{M}_3$
-0.020019
$\mathbb{M}_4$
1.035525
$\mathbb{M}_5$


1.098612
$\mathbb{M}_0$
0.1682361
$\mathbb{M}_1$
-0.2630465
$\mathbb{M}_2$
1.202774
$\mathbb{M}_3$
-0.020019
$\mathbb{M}_4$
1.035525
$\mathbb{M}_5$
q
Weighted Average
(using confidence factors α)
t

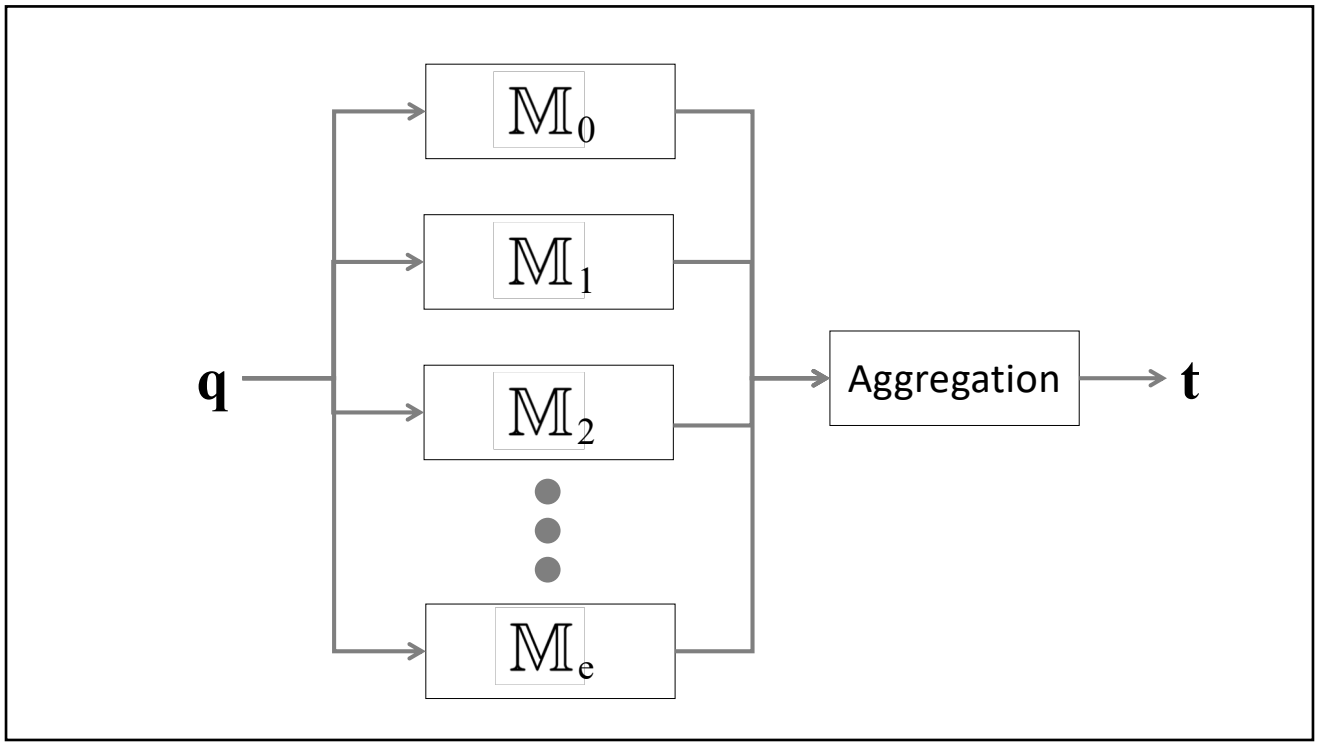

1.098612
$\mathbb{M}_0$
0.1682361
$\mathbb{M}_1$
-0.2630465
$\mathbb{M}_2$
1.202774
$\mathbb{M}_3$
-0.020019
$\mathbb{M}_4$
1.035525
$\mathbb{M}_5$
q
Weighted Average
(using confidence factors α)
t
$t = sign\left(\sum_{\mathbb{M}_i \in \mathbb{M}} \alpha_i \mathbb{M}_i(q)\right)$

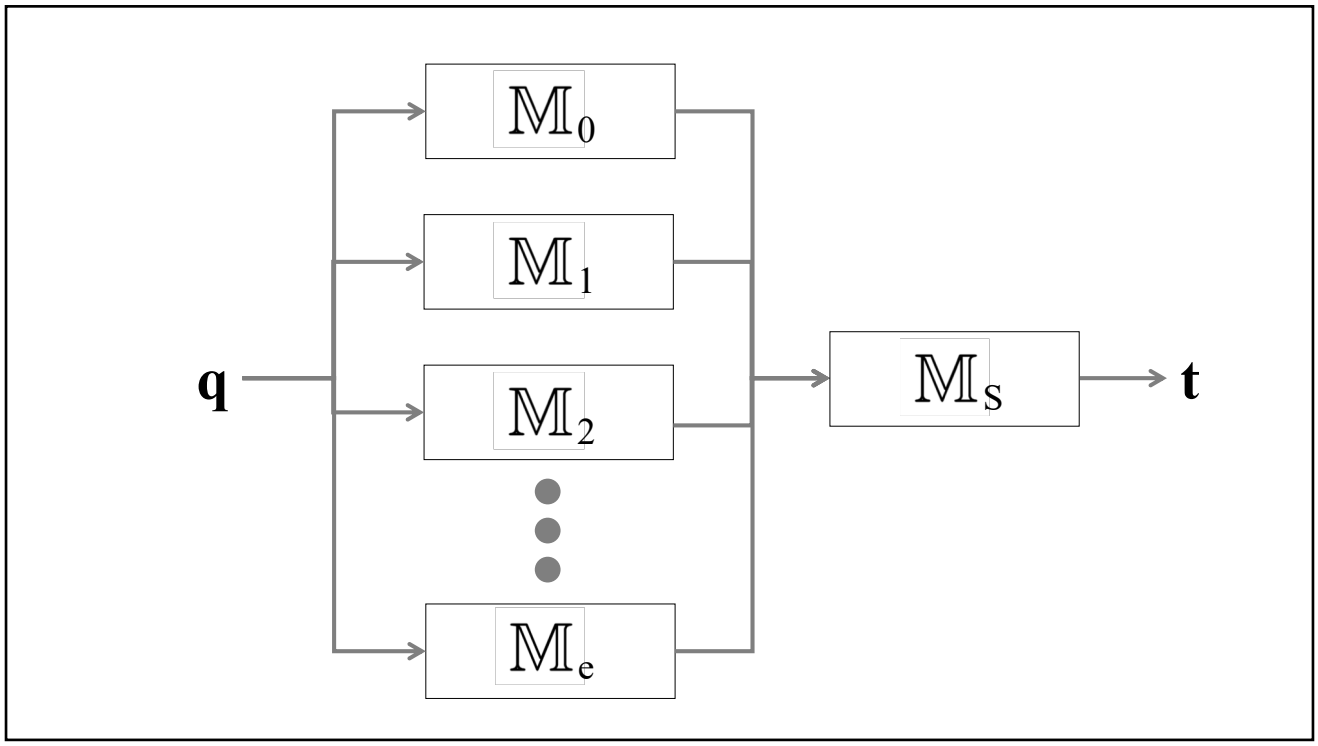
STACKING

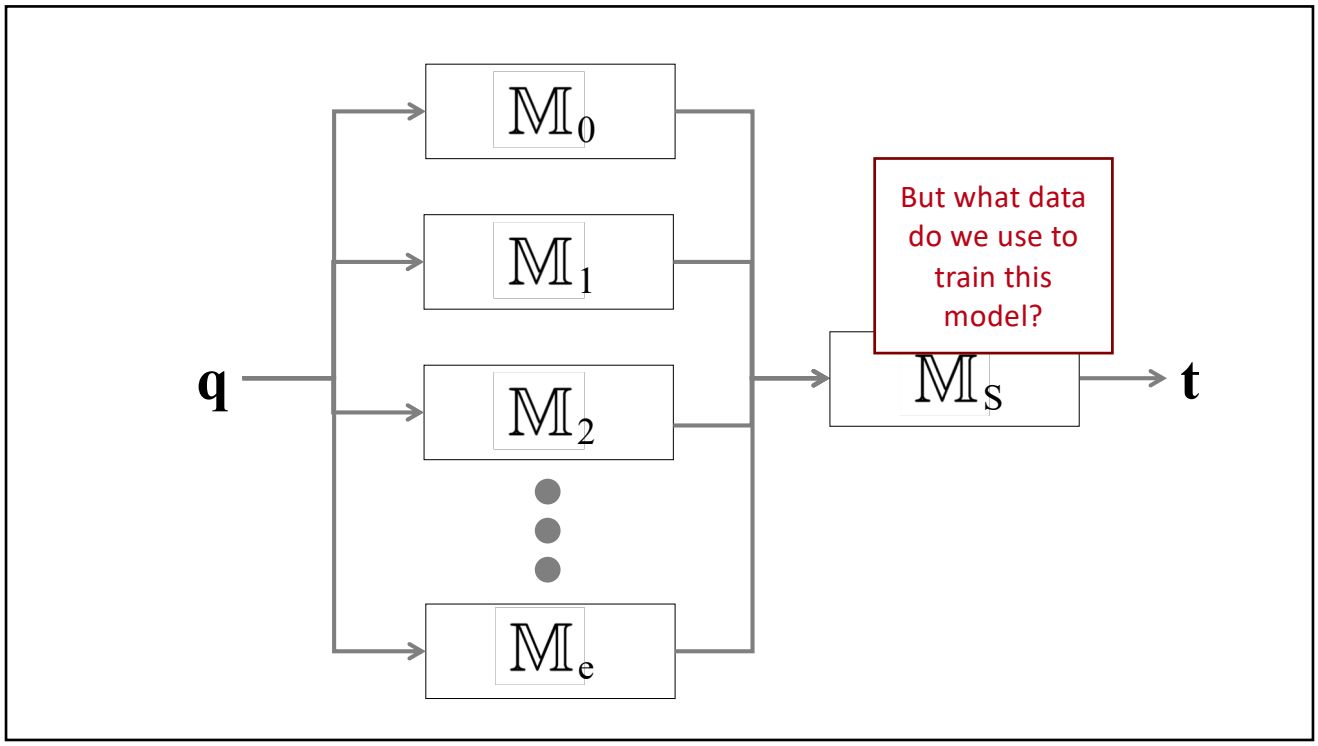
Stacking

Stacking ensembles use a machine learning model to combine the outputs of the base models in an ensemble

- Can be more effective than simple majority voting or weighted voting
- Requires new datasets to be generated


$\mathbb{M}_0$
$\mathbb{M}_1$
$\mathbb{M}_2$
$\mathbb{M}_e$
q
Aggregation
t


$\mathbb{M}_0$
$\mathbb{M}_1$
$\mathbb{M}_2$
$\mathbb{M}_e$
q
$\mathbb{M}_S$
t



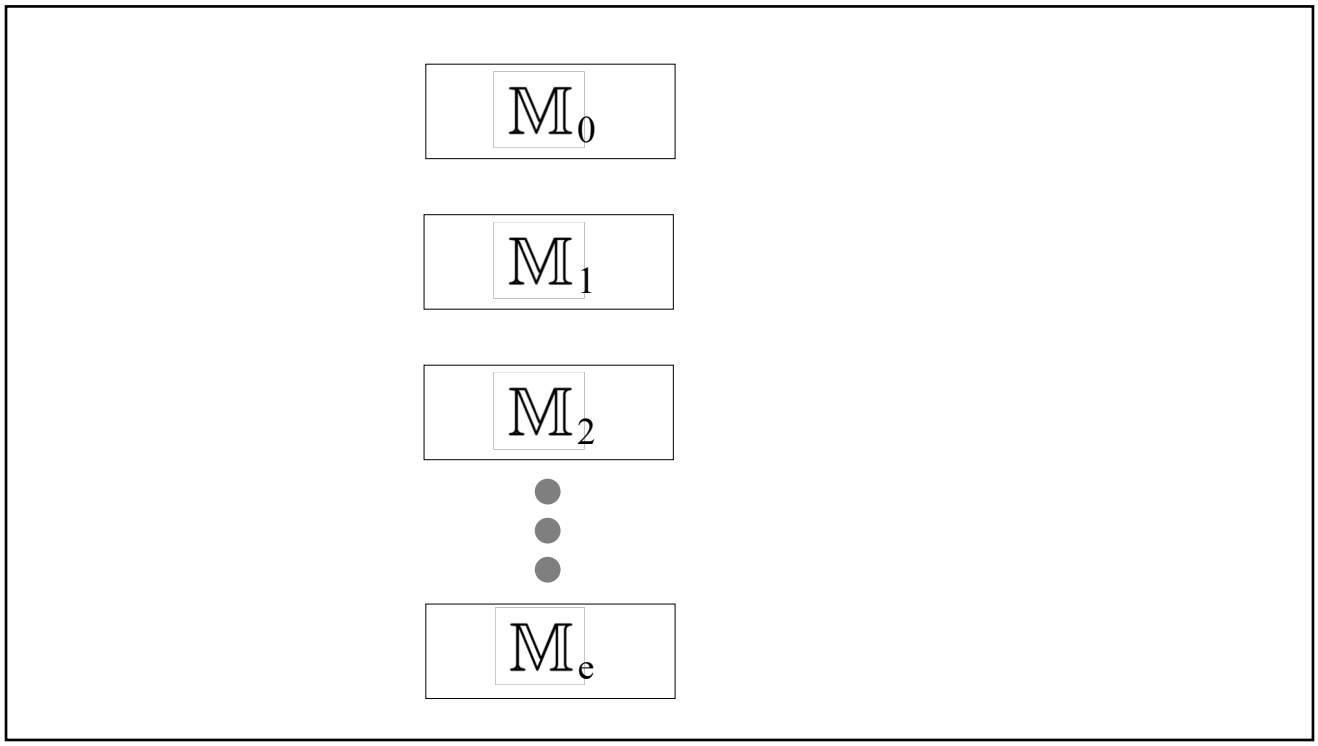
	$\mathbb{M}_0$	$\mathbb{M}_1$	$\mathbb{M}_2$	$\mathbb{M}_3$	•••	$\mathbb{M}_e$	Target
d_0	True	False	True	True		False	True
d_1	False	False	False	False		True	False
	⋮						
d_n	True	True	True	False		False	False

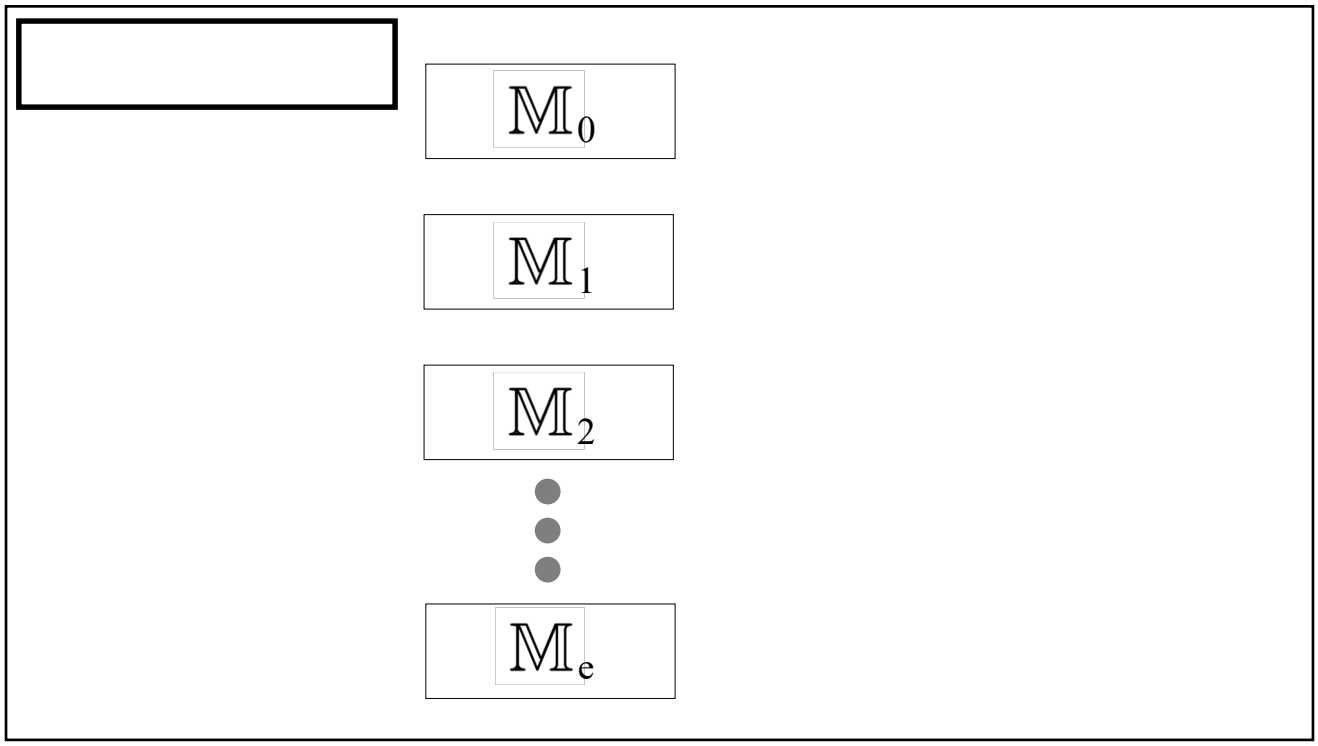
	$\mathbb{M}_0$	$\mathbb{M}_1$	$\mathbb{M}_2$	$\mathbb{M}_3$	•••	$\mathbb{M}_e$	Target
d_0	0.81	0.22	0.76	0.91		0.11	True
d_1	0.38	0.41	0.29	0.38		0.55	False
				⋮			
d_n	0.99	0.76	0.54	0.44		0.38	False

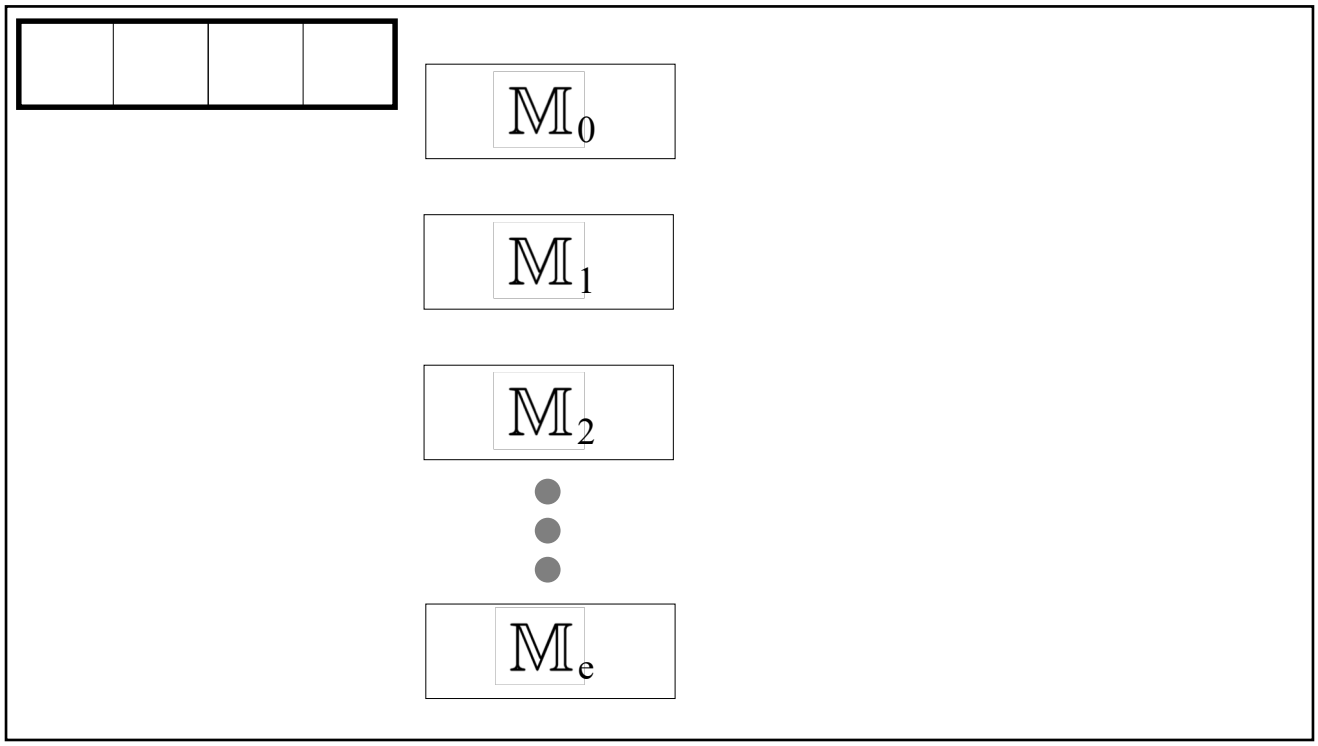
Stacking

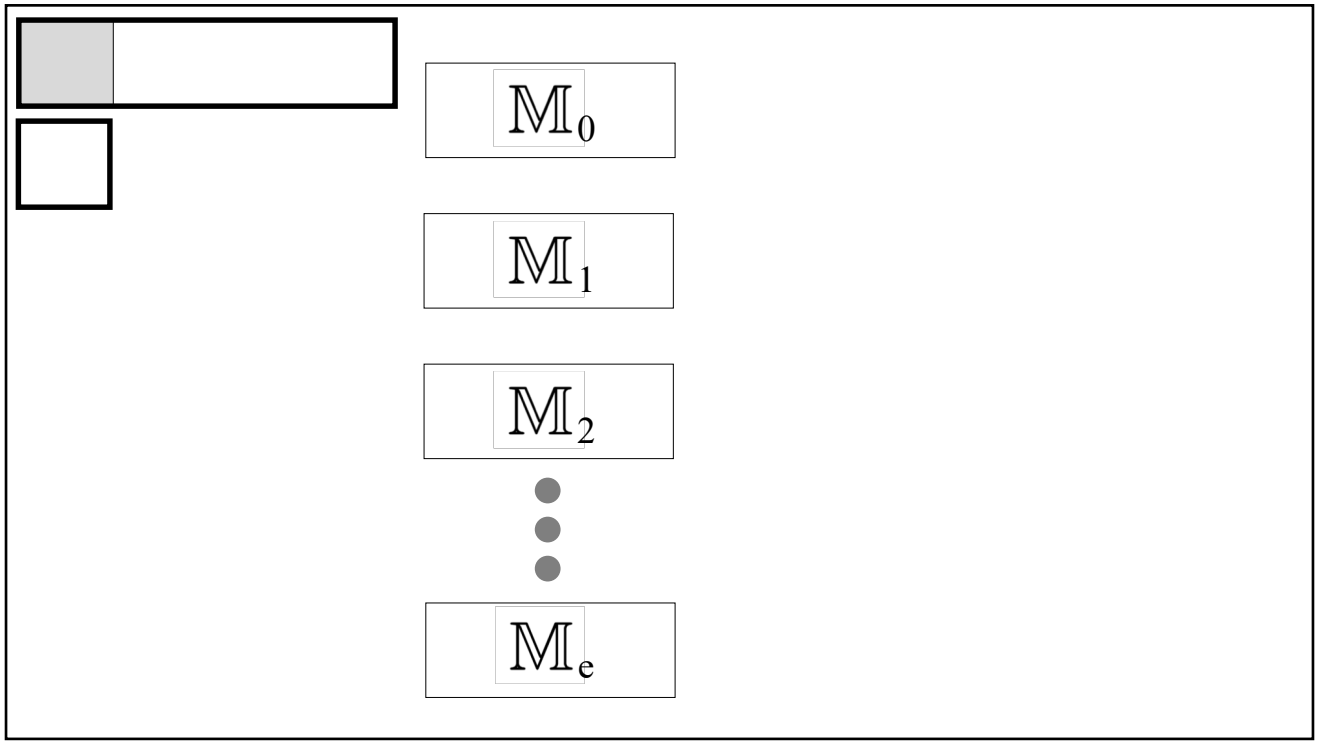
If exactly the same data used to train the base learners is also used to train the stacking model there is a serious risk of overfitting

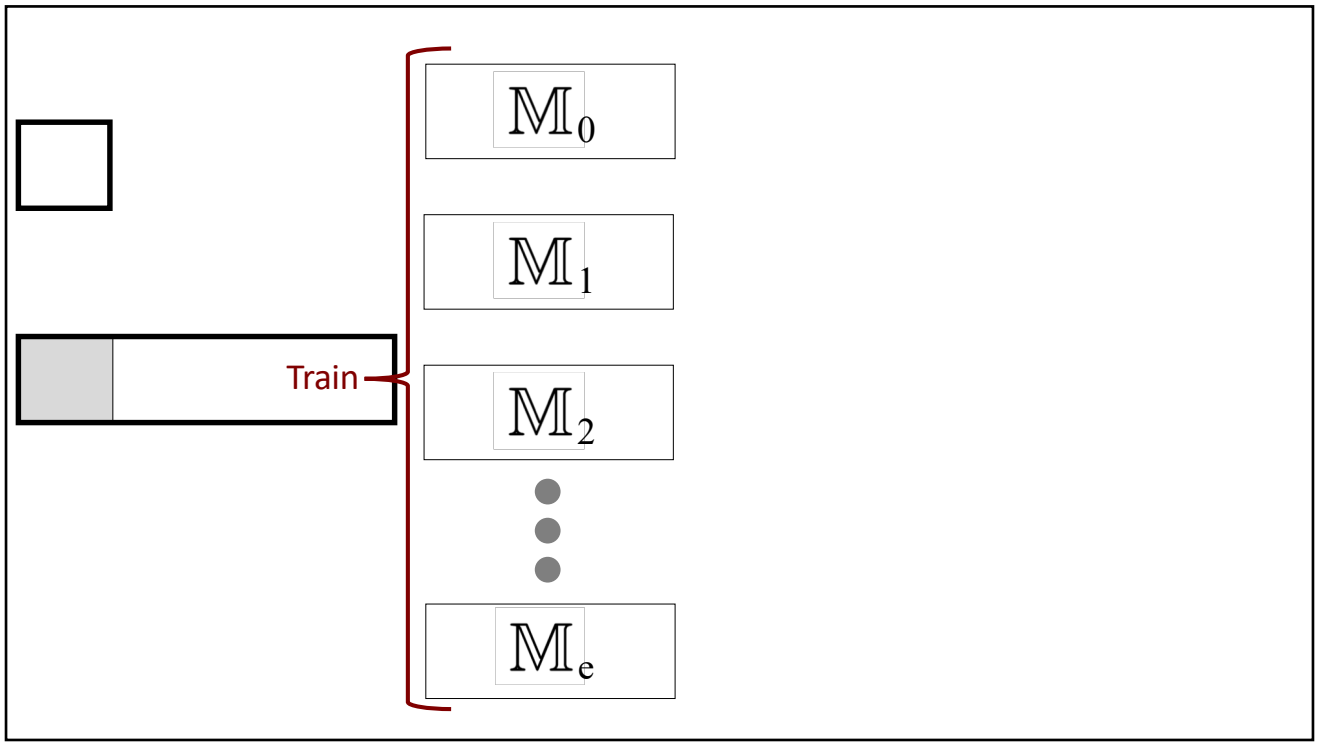
Common to use a k-fold cross validation scheme to gerneate the stacked level training set

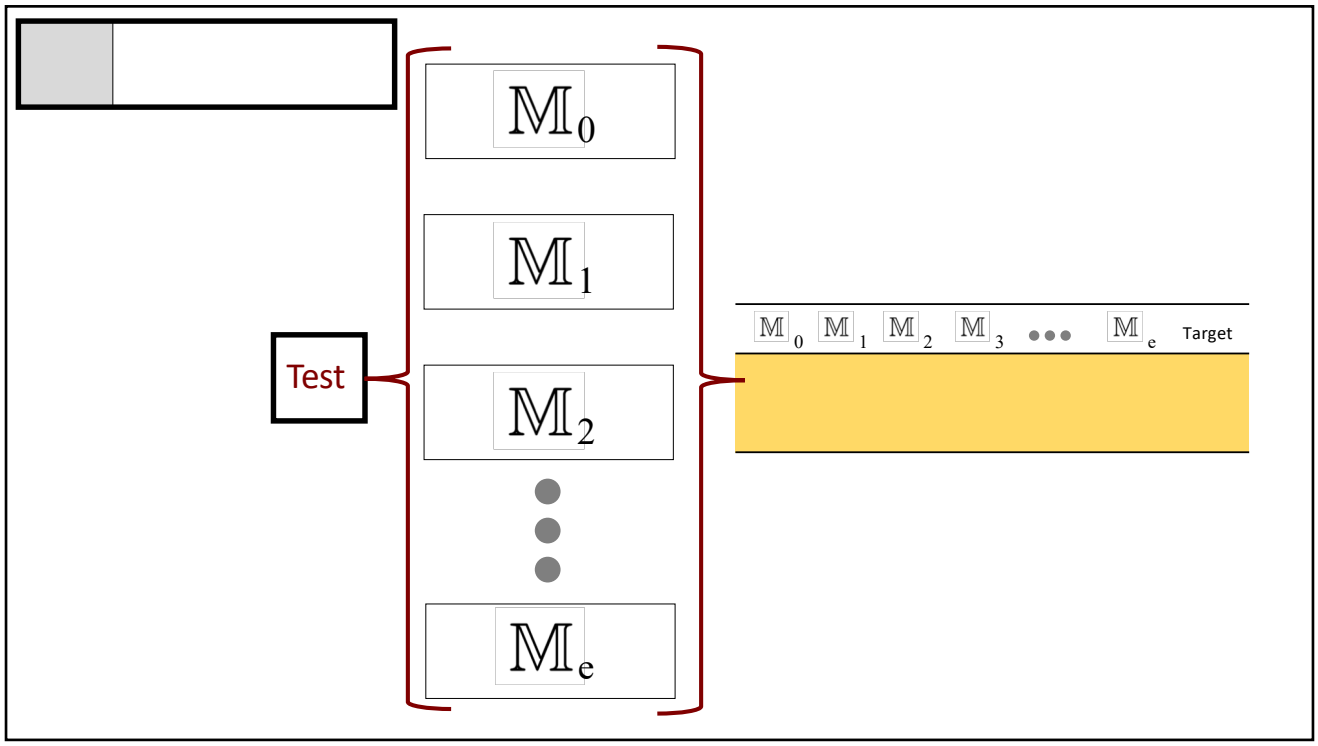

$\mathbb{M}_0$
$\mathbb{M}_1$
$\mathbb{M}_2$
$\mathbb{M}_e$

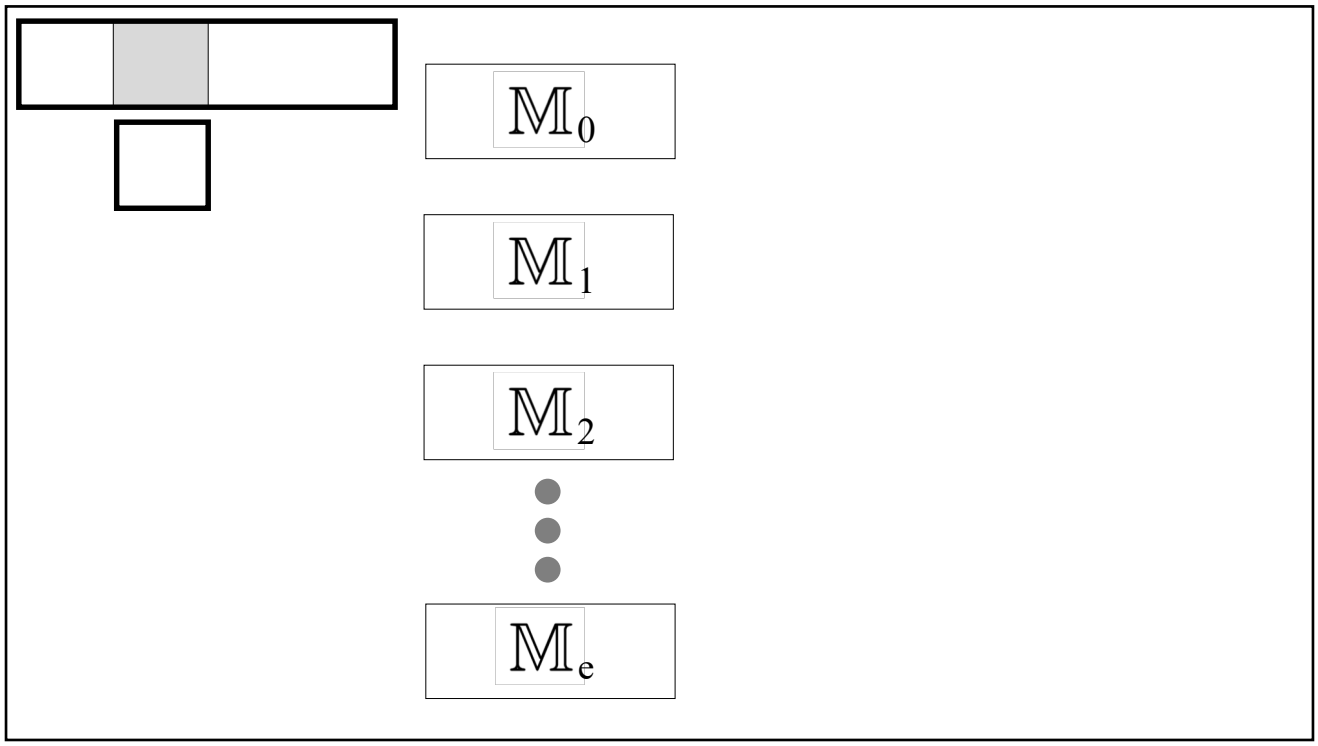

$\mathbb{M}_0$
$\mathbb{M}_1$
$\mathbb{M}_2$
$\mathbb{M}_e$

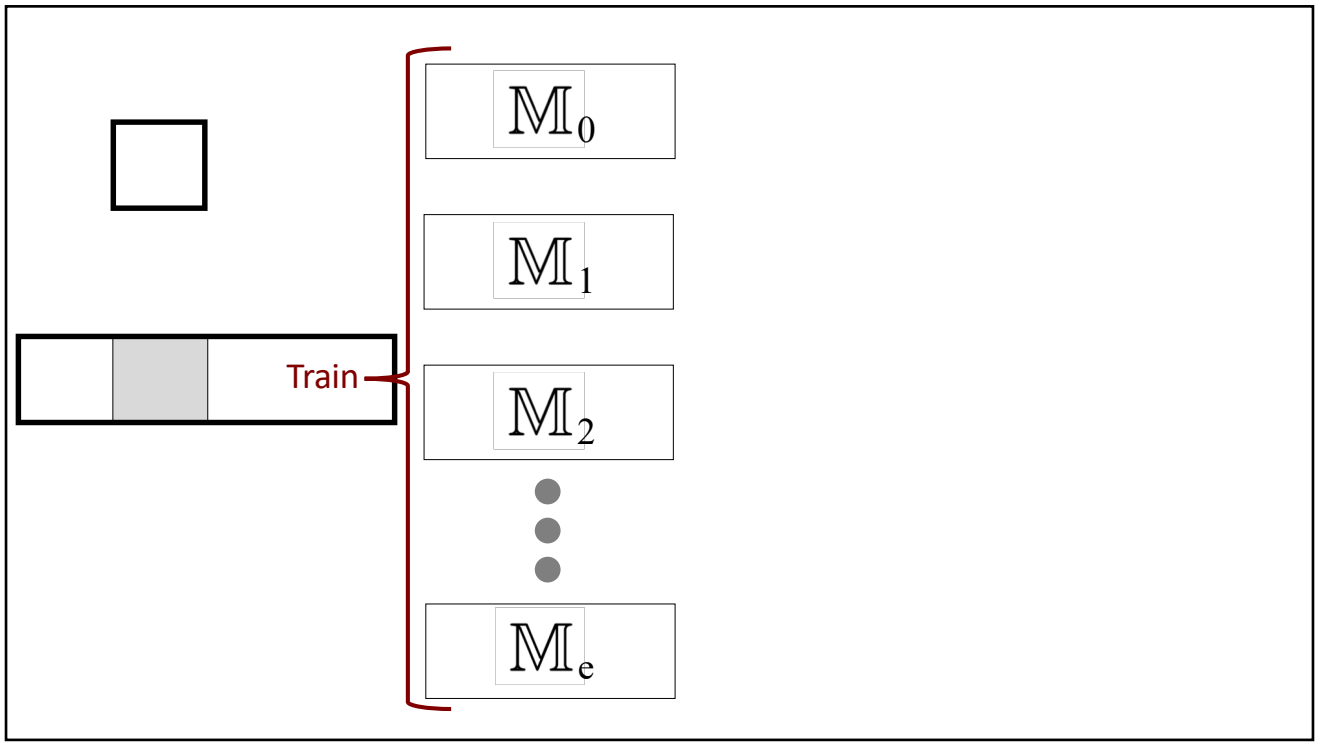

$\mathbb{M}_0$
$\mathbb{M}_1$
$\mathbb{M}_2$
$\mathbb{M}_e$

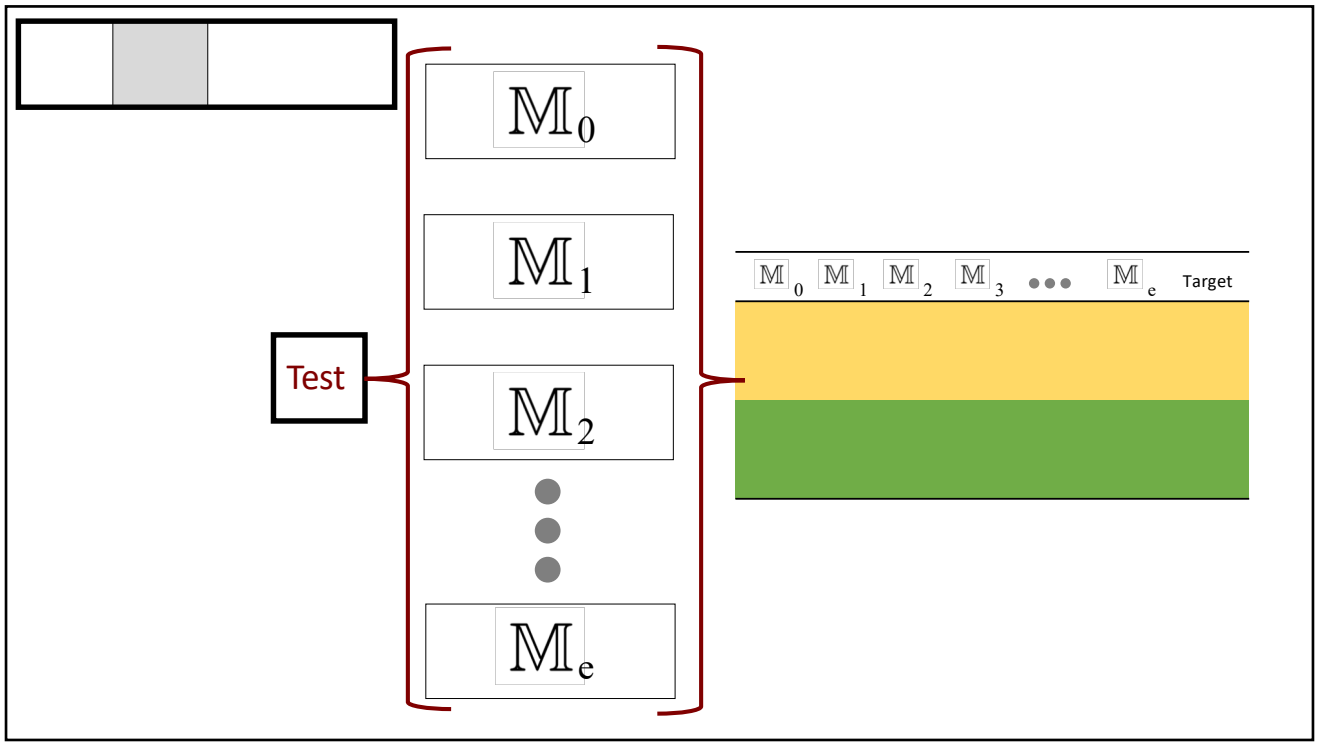

$\mathbb{M}_0$
$\mathbb{M}_1$
$\mathbb{M}_2$
$\mathbb{M}_e$

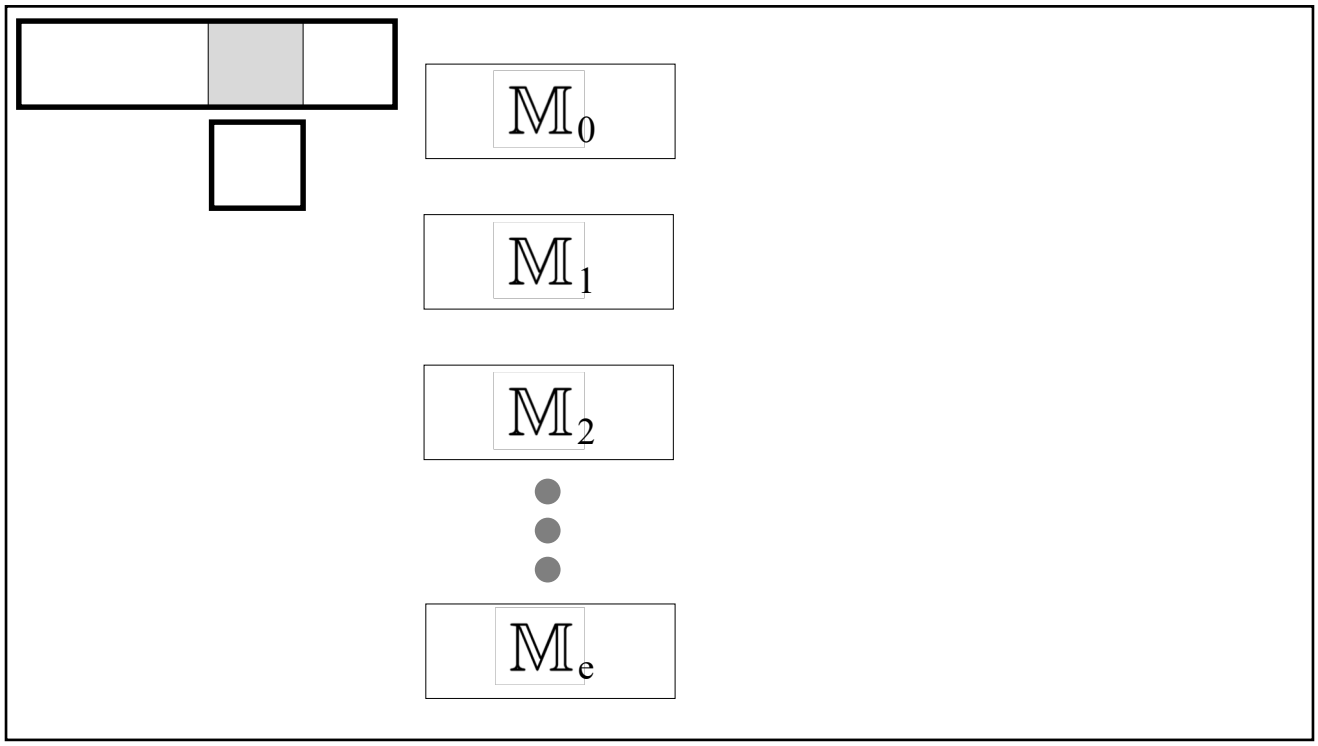

$\mathbb{M}_0$
$\mathbb{M}_1$
Train
$\mathbb{M}_2$
$\mathbb{M}_e$

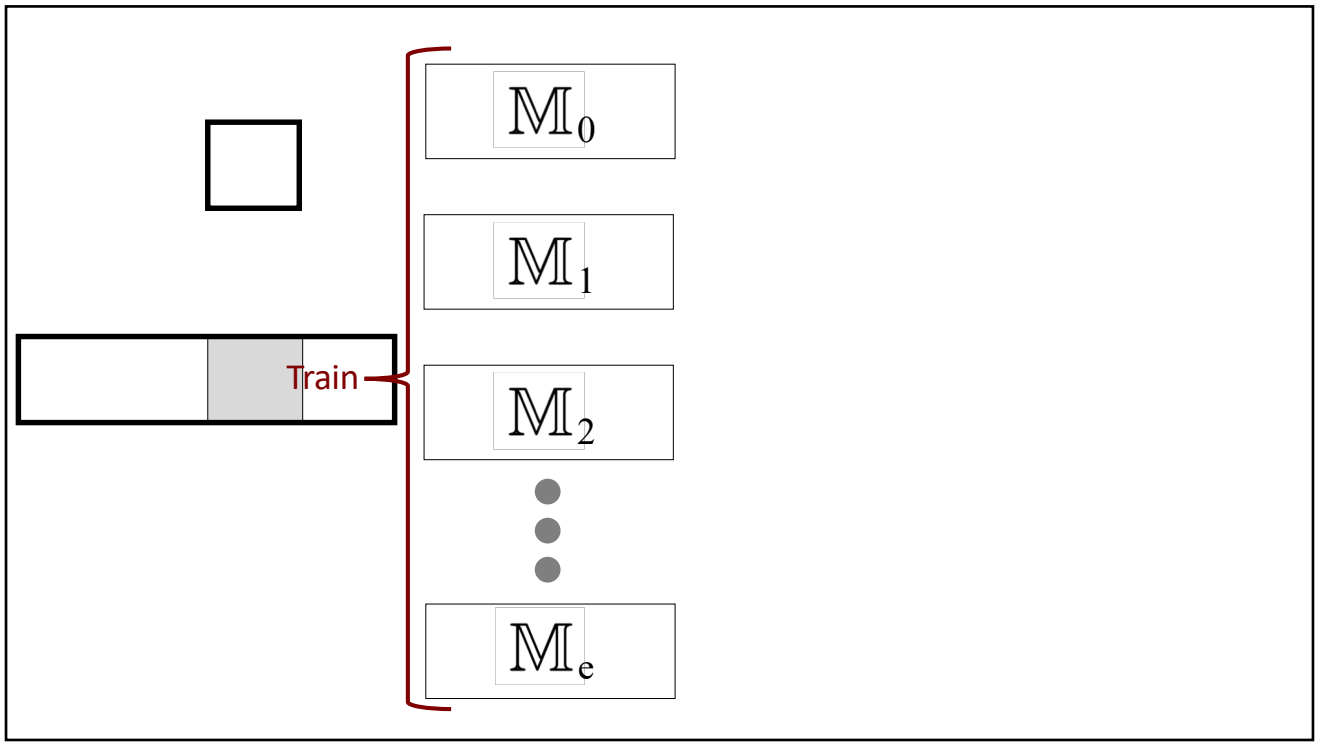

$\mathbb{M}_0$
$\mathbb{M}_1$
Test
$\mathbb{M}_2$
$\mathbb{M}_e$
$\mathbb{M}_0$ $\mathbb{M}_1$ $\mathbb{M}_2$ $\mathbb{M}_3$ ●●● $\mathbb{M}_e$ Target

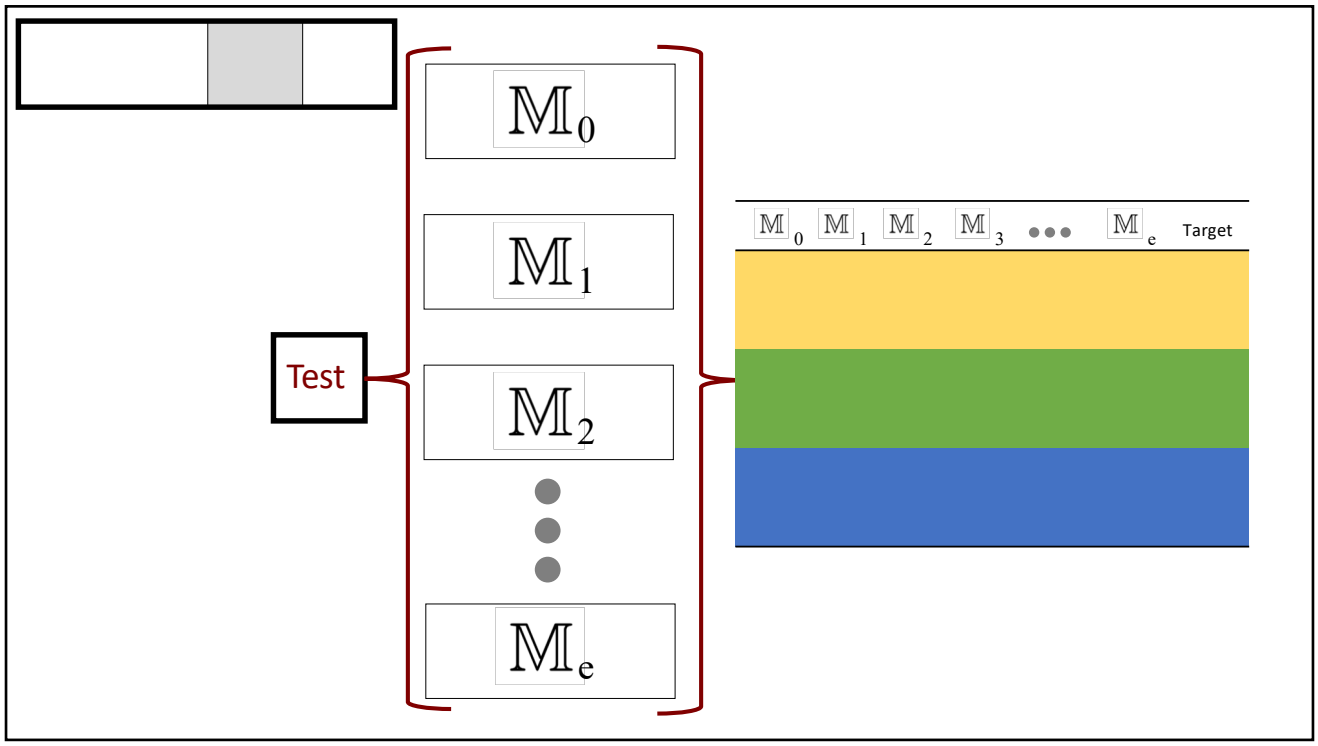

$\mathbb{M}_0$
$\mathbb{M}_1$
$\mathbb{M}_2$
$\mathbb{M}_e$

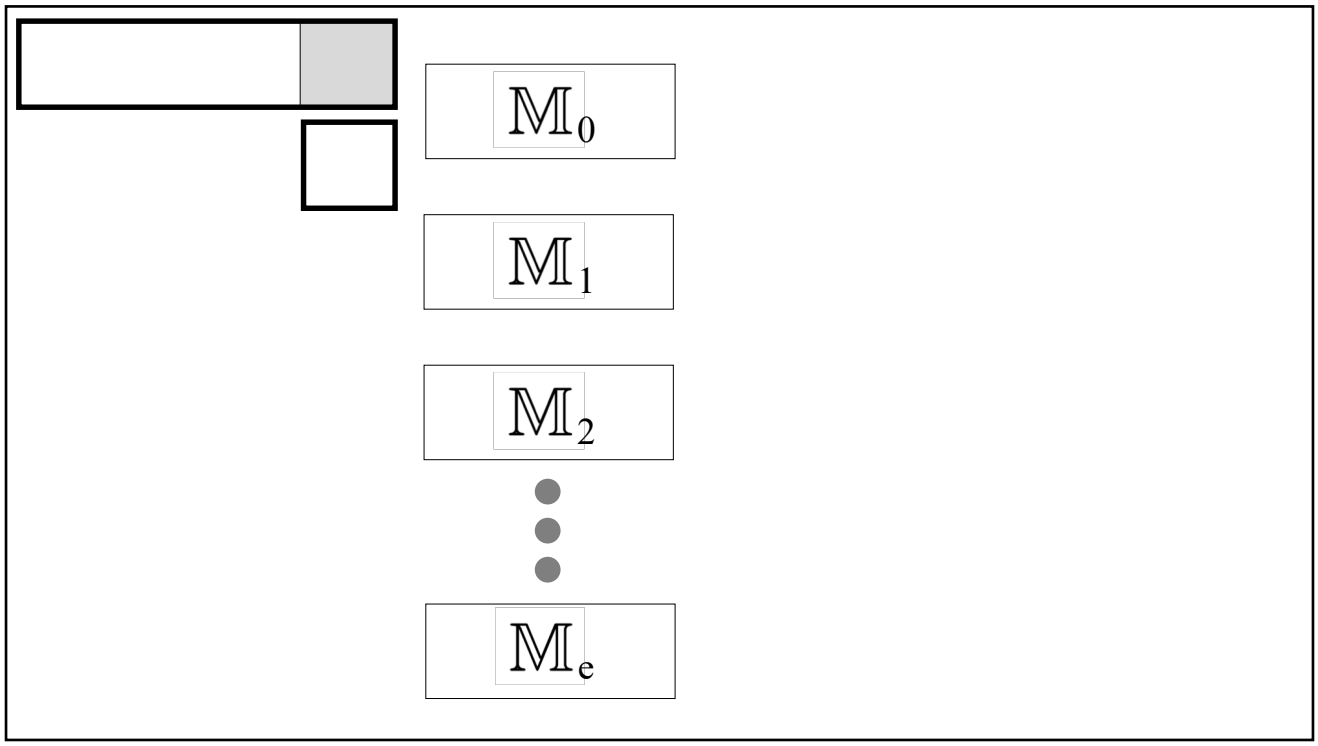

Train
$\mathbb{M}_0$
$\mathbb{M}_1$
$\mathbb{M}_2$
$\mathbb{M}_e$

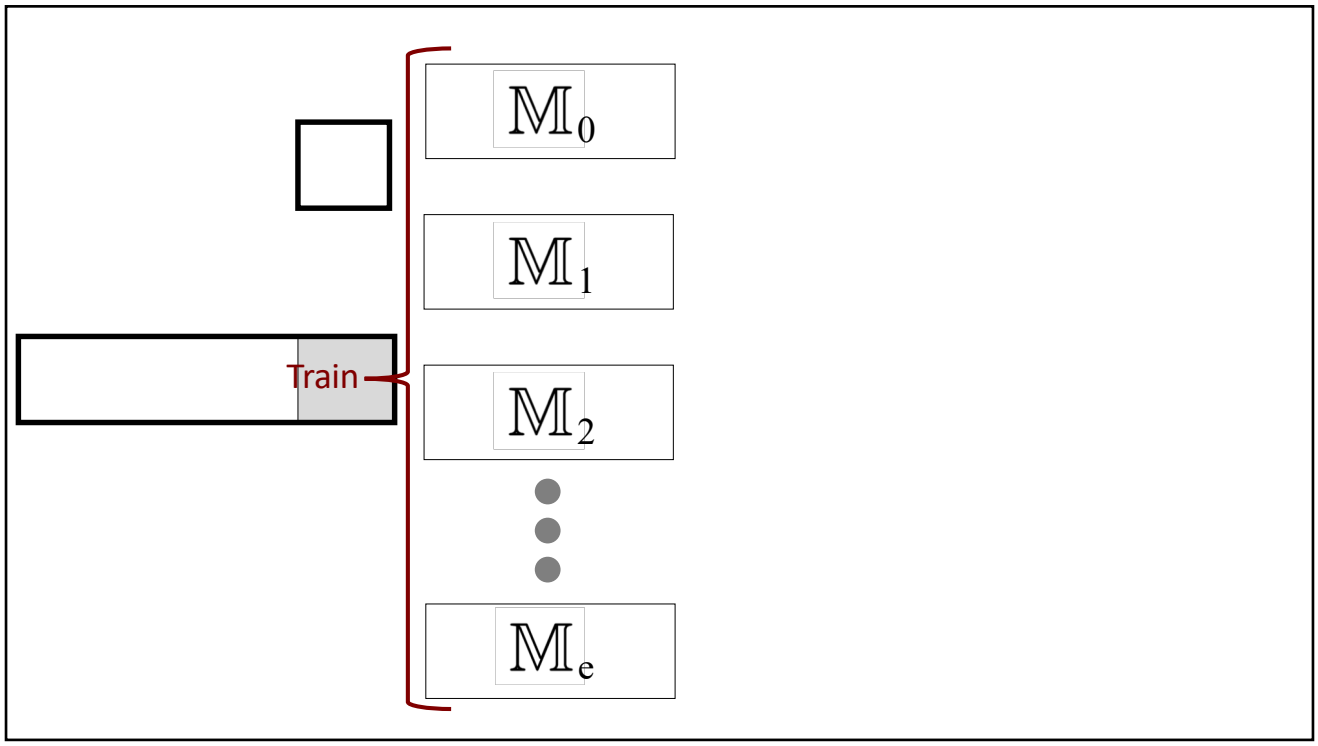

$\mathbb{M}_0$
$\mathbb{M}_1$
Test
$\mathbb{M}_2$
$\mathbb{M}_e$
$\mathbb{M}_0$ $\mathbb{M}_1$ $\mathbb{M}_2$ $\mathbb{M}_3$ $\mathbb{M}_e$ Target

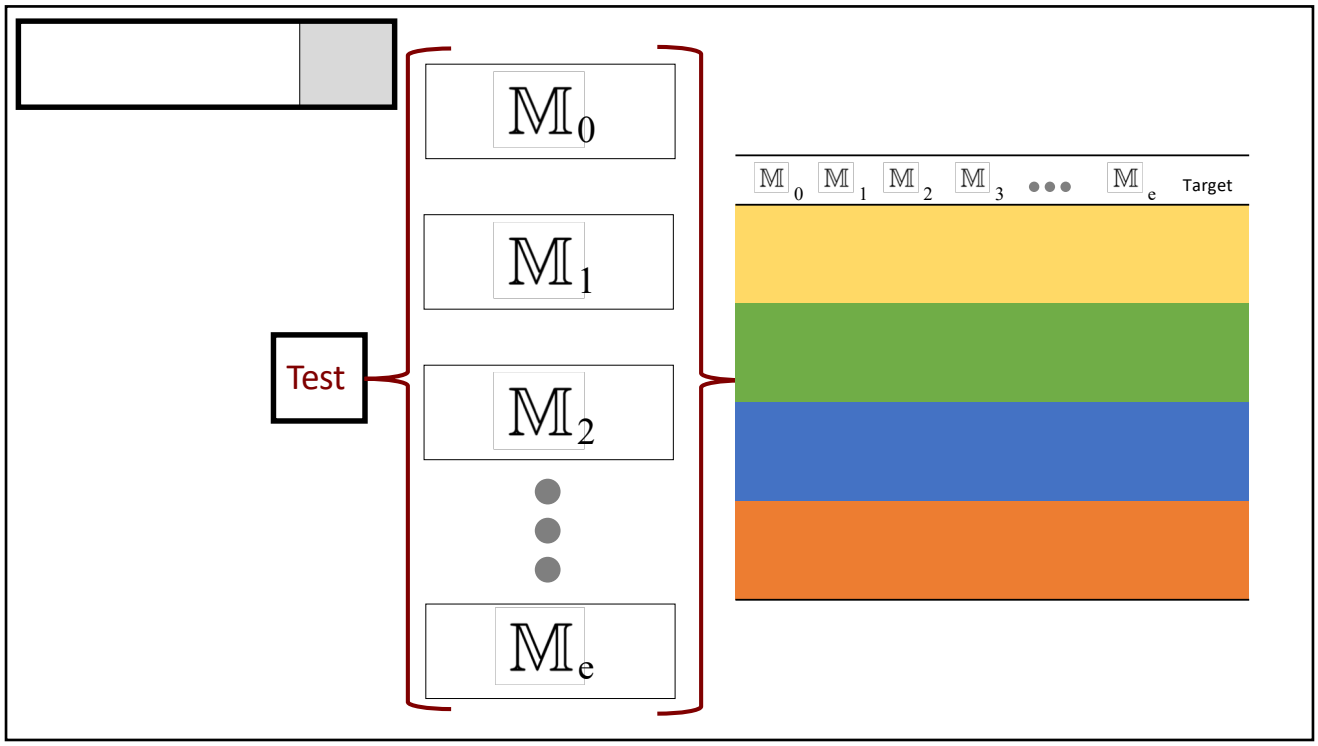

$\mathbb{M}_0$
$\mathbb{M}_1$
$\mathbb{M}_2$
$\mathbb{M}_e$

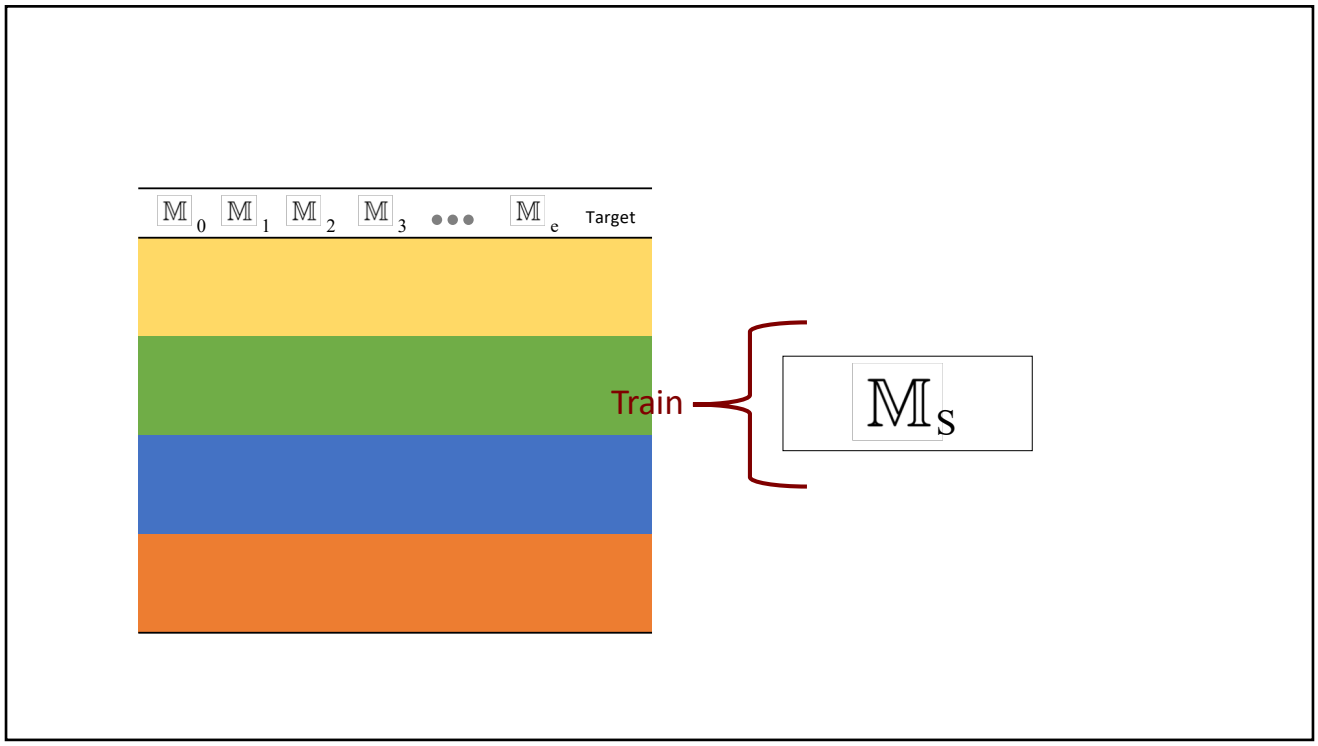

$\mathbb{M}_0$
$\mathbb{M}_1$
Train
$\mathbb{M}_2$
$\mathbb{M}_e$

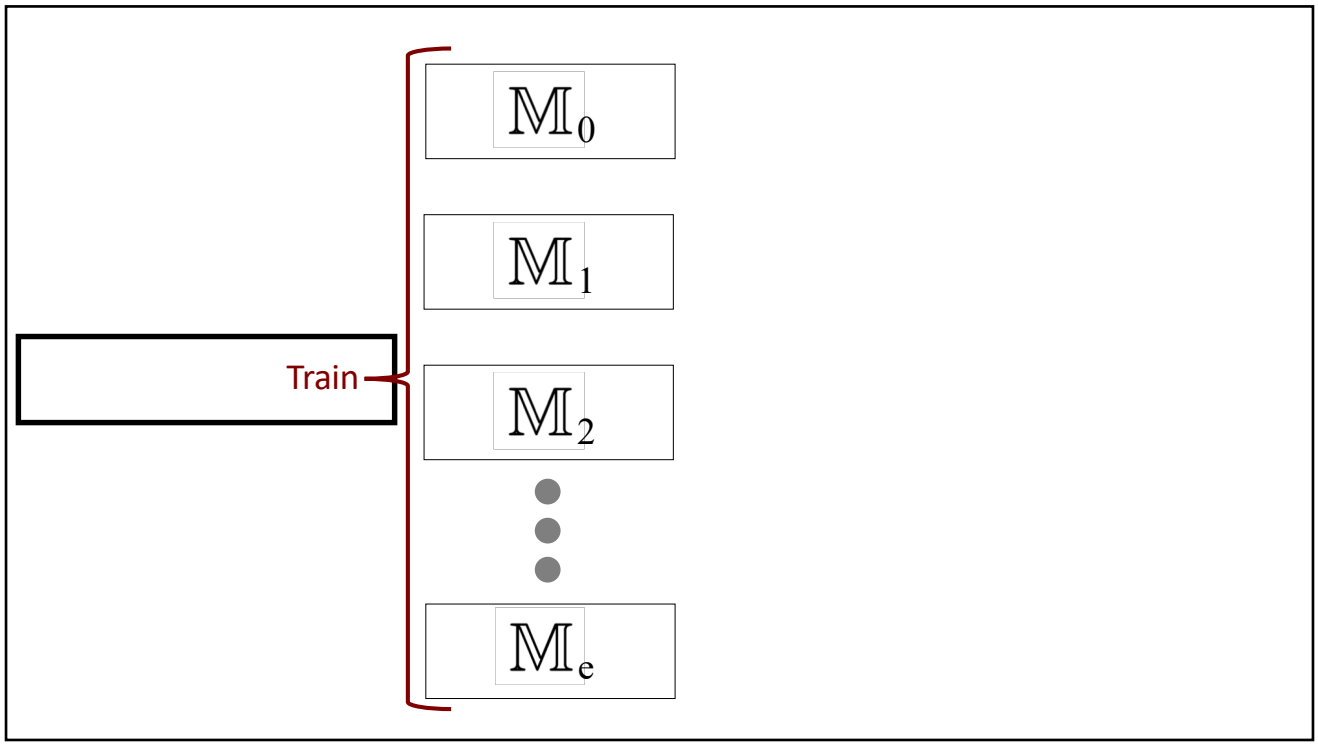

$\mathbb{M}_0$
$\mathbb{M}_1$
Test
$\mathbb{M}_2$
$\mathbb{M}_e$
$\mathbb{M}_0$ $\mathbb{M}_1$ $\mathbb{M}_2$ $\mathbb{M}_3$ ••• $\mathbb{M}_e$ Target

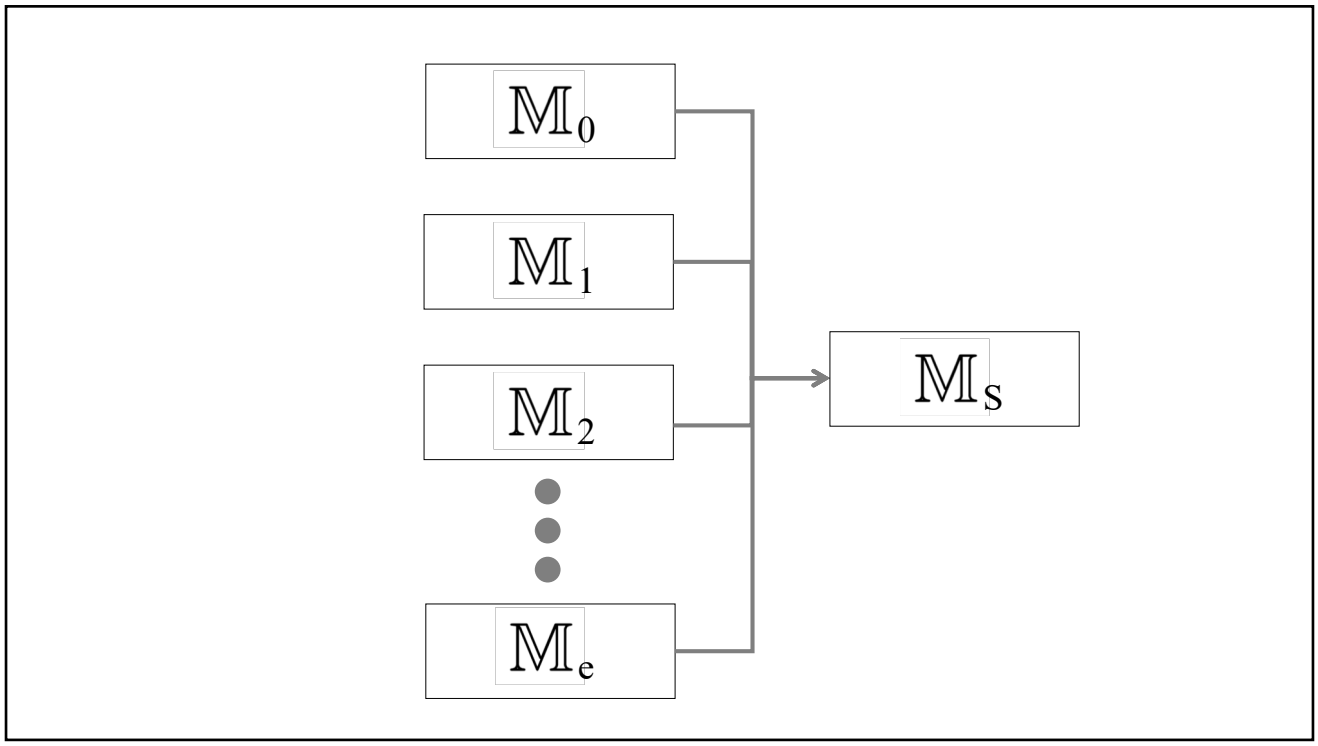

$\mathbb{M}_0$
$\mathbb{M}_1$
$\mathbb{M}_2$
$\mathbb{M}_e$

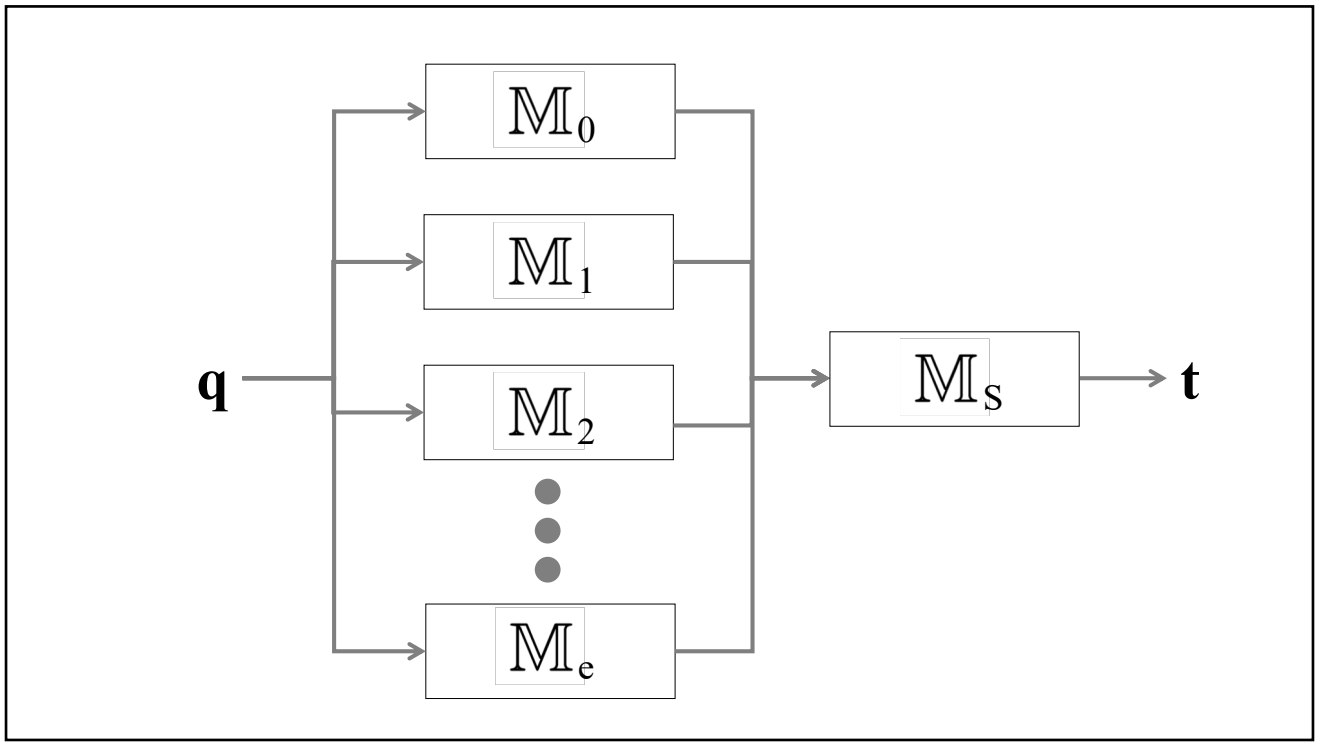

$\mathbb{M}_0$
$\mathbb{M}_1$
Train
$\mathbb{M}_2$
$\mathbb{M}_e$


$\mathbb{M}_0$
$\mathbb{M}_1$
Test
$\mathbb{M}_2$
$\mathbb{M}_e$
$\mathbb{M}_0$ $\mathbb{M}_1$ $\mathbb{M}_2$ $\mathbb{M}_3$ ●●● $\mathbb{M}_e$ Target


$\mathbb{M}_0$ $\mathbb{M}_1$ $\mathbb{M}_2$ $\mathbb{M}_3$ ●●● $\mathbb{M}_e$ Target
Train
$\mathbb{M}_S$


$\mathbb{M}_0$
$\mathbb{M}_1$
Train
$\mathbb{M}_2$
$\mathbb{M}_e$


$\mathbb{M}_0$
$\mathbb{M}_1$
$\mathbb{M}_2$
$\mathbb{M}_e$
$\mathbb{M}_S$

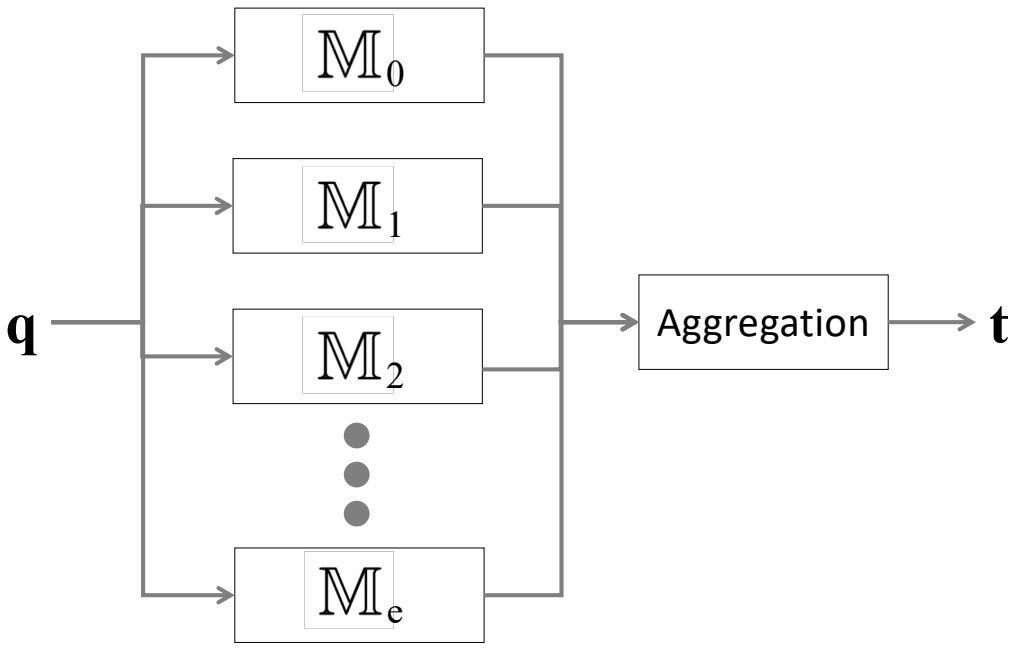


Stacking

It is very common to use **heterogenous ensembles** with stacking

Stacking takes a bit of work, but can be effective

GRADIENT BOOSTING



Gradient Boosting

Gradient boosting creates an ensemble model by iteratively adding learners - similar to AdaBoost

Gradient boosting is more aggessive fitting each new model directly to the errors of the ensemble (as constituted up to the current iteration) rather then to a weighted dataset which is more subtle

Gradient Boosting

Gradient boosting is best explained in the context of predicting a continuous target

In a regression task we are trying to predict a continuous target and the goal of training is to minimise some measure of error; e.g., the mean squared error:

$$MSE = \frac{\sum_{i=1}^{n} (t_i - \mathbb{M}(\mathbf{d}_i))^2}{n}$$

Gradient Boosting

At each iteration gradient boosting assumes we already have a model that can make predictions (this model can be very weak)

For example, in the first iteration this model may simply predict the mean of the target

$$\mathbb{M}\left(\mathbf{d}_i\right) = \frac{\sum_{i=1}^{n} t_i}{n}$$

Gradient Boosting

Gradient boosting improves this existing model by adding a new model that reduces the error of the existing model

Gradient Boosting

Iteration	Ensemble Model
1	$\mathbb{M}_1 = \frac{\sum_{i=1}^{n} t_i}{n}$
2	$\mathbb{M}_2 = \mathbb{M}_1 + \mathbb{M}_{iter2}$
3	$\mathbb{M}_3 = \mathbb{M}_2 + \mathbb{M}_{iter3}$
	$\ldots$
n	$\mathbb{M}_n = \mathbb{M}_{n-1} + \mathbb{M}_{iterN}$

Gradient Boosting

The question is how to define the model that we add to the existing model

The solution adopted by gradient boosting is based on the intuition that the perfect model to add would be the model that made the predictions for the total ensemble correct:

$$\mathbb{M}_n(\mathbf{d}_i) = \mathbb{M}_{n-1}(\mathbf{d}_i) + \mathbb{M}_{iterN}(\mathbf{d}_i) = t_i$$

Gradient Boosting

From the above equation we can see that the best model to fit would be the model that predicts the difference between the old models prediction and the true prediction:

$$\mathbb{M}_{iterN}(\mathbf{d}_i) = t_i - \mathbb{M}_{n-1}(\mathbf{d}_i)$$

Gradient Boosting

So gradient boosting trains the new model to add to the ensemble by training the model to predict the errors (in regression terms the residuals) of the old model

Gradient Boosting Example

Consider a simple initial dataset

ID	N descriptive features	Target
1	...	10
2	...	15
3	...	6
4	...	18
	...	

Gradient Boosting Example

Assuming that the first model in the ensemble predicted the average of the target values (**9**) the residuals for this first model would be

ID	N descriptive features	Target	$\mathbb{M}(\mathbf{d}_i)$	$Residuals_1$
1	...	10	9	1
2	...	15	9	6
3	...	6	9	-3
4	...	18	9	9
	...			

Gradient Boosting Example

Assuming that the first model in the ensemble predicted the average of the target the residuals for this first model would be

ID	N descriptive features	Target	$\mathbb{M}(\mathbf{d}_i)$	$Residuals_1$
1	...	10	9	1
2	...	15	9	6
3	...	6	9	-3
4	...	18	9	9
	...			

Gradient Boosting Example

So the second model $\mathbb{M}_2$ in the ensemble would be trained on the following dataset:

ID	N descriptive features	$Residuals_1$
1	...	1
2	...	6
3	...	-3
4	...	9
	...	

Gradient Boosting Example

And the prediction used to calculate the residuals for the next iteration would be defined as:

ID	N descriptive features	Target	$Residuals_2$
1	...	10	10 - $(\mathbb{M}_1(\mathbf{d}_i) + \mathbb{M}_2(\mathbf{d}_i))$
2	...	15	15 - $(\mathbb{M}_1(\mathbf{d}_i) + \mathbb{M}_2(\mathbf{d}_i))$
3	...	6	6 - $(\mathbb{M}_1(\mathbf{d}_i) + \mathbb{M}_2(\mathbf{d}_i))$
4	...	18	18 - $(\mathbb{M}_1(\mathbf{d}_i) + \mathbb{M}_2(\mathbf{d}_i))$
	...		

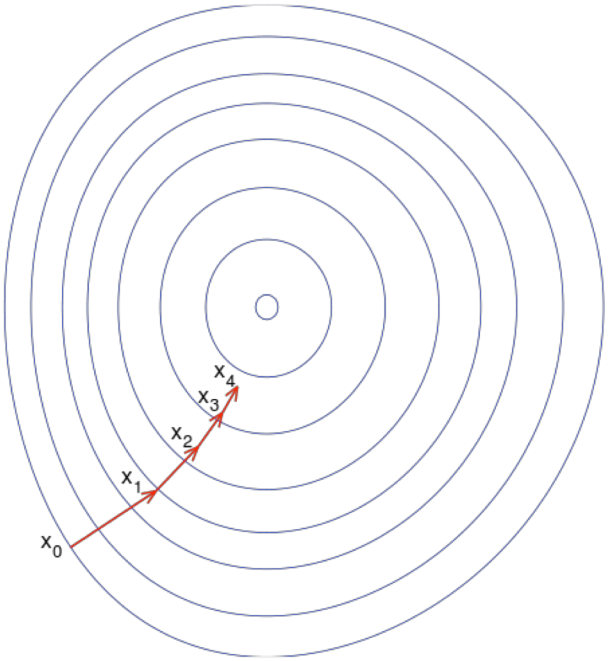
Gradient Boosting Example

Gradient boosting is called gradient boosting because we can treat the residuals

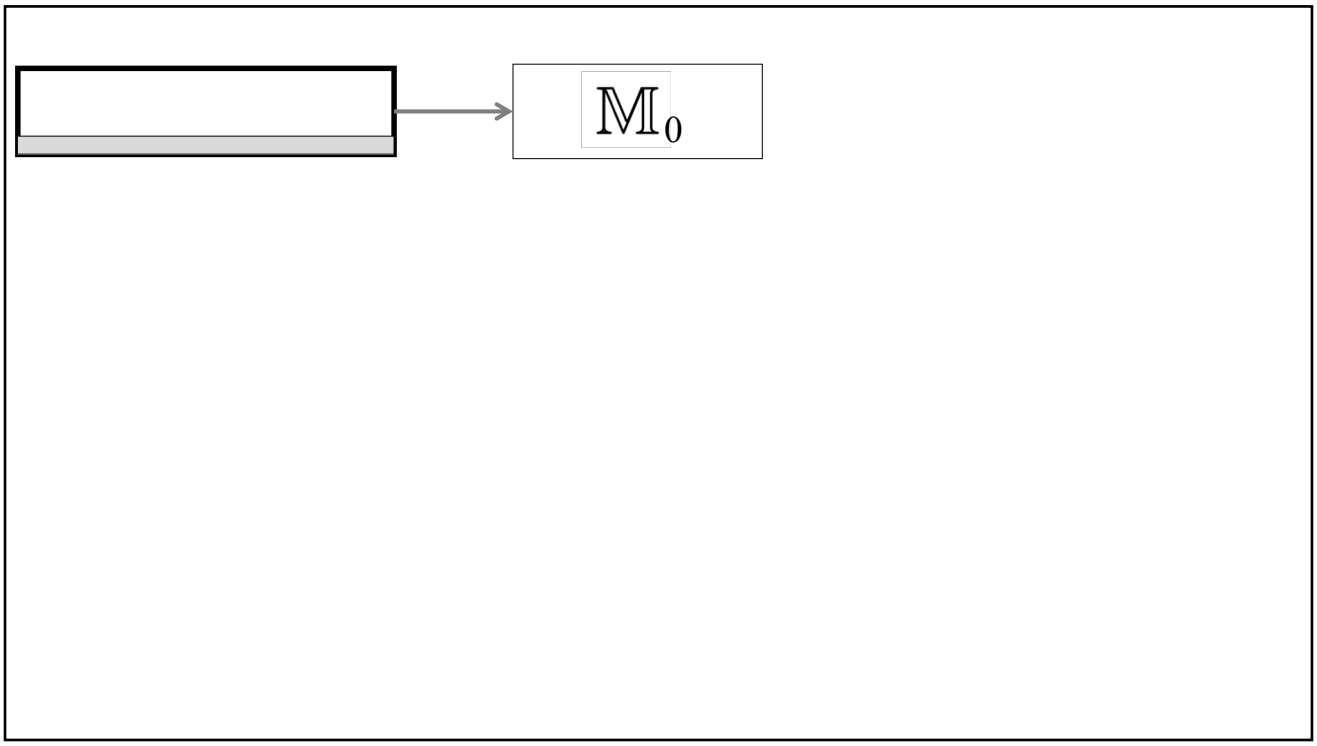
$$t_i - \mathbb{M}_{n-1}(\mathbf{d}_i)$$

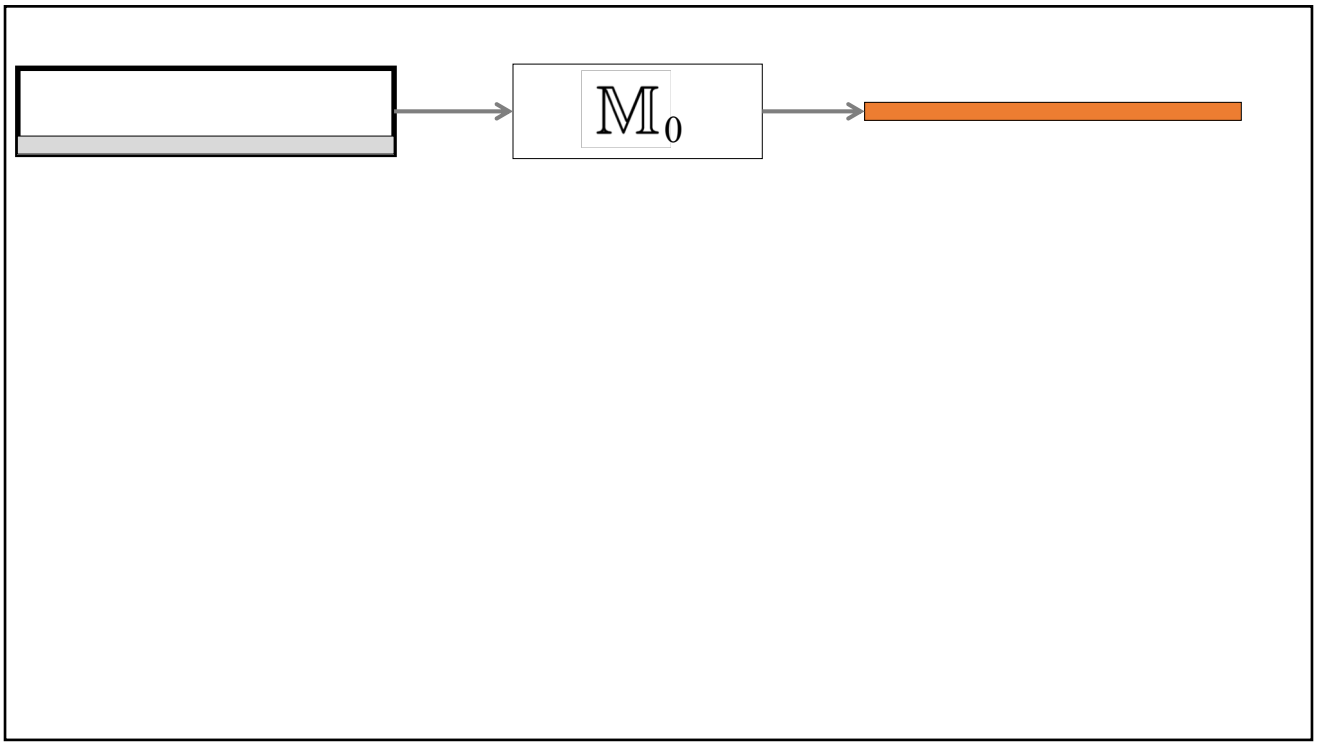
as the negative gradients of the squared error loss function

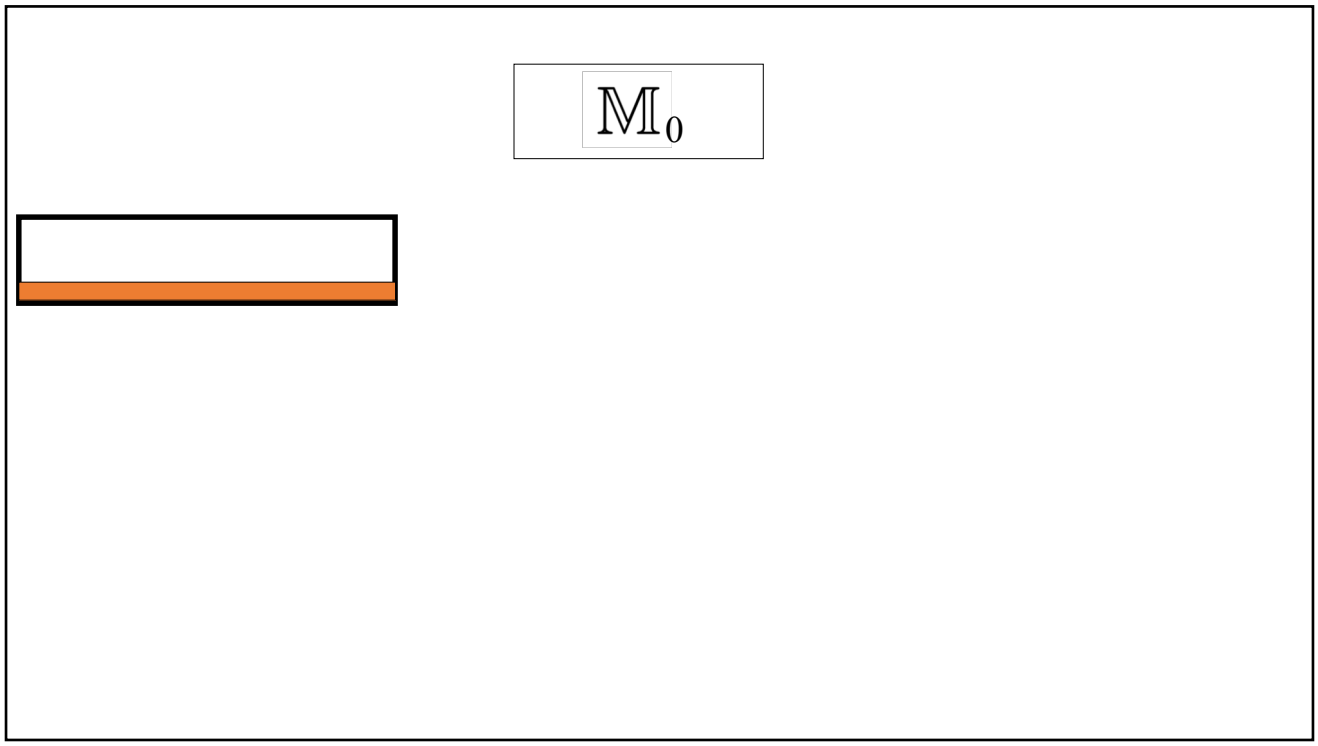
So, under the hood gradient boosting is essentially doing gradient descent on an error surface

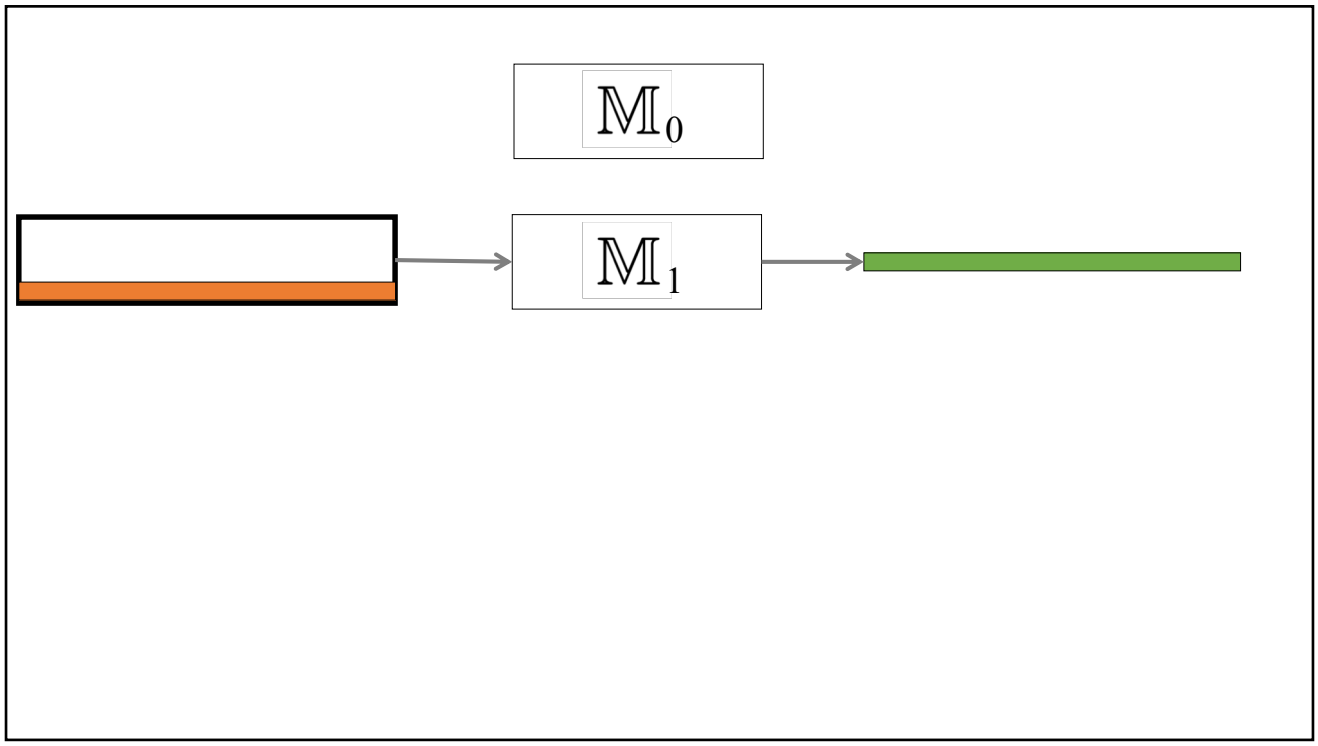


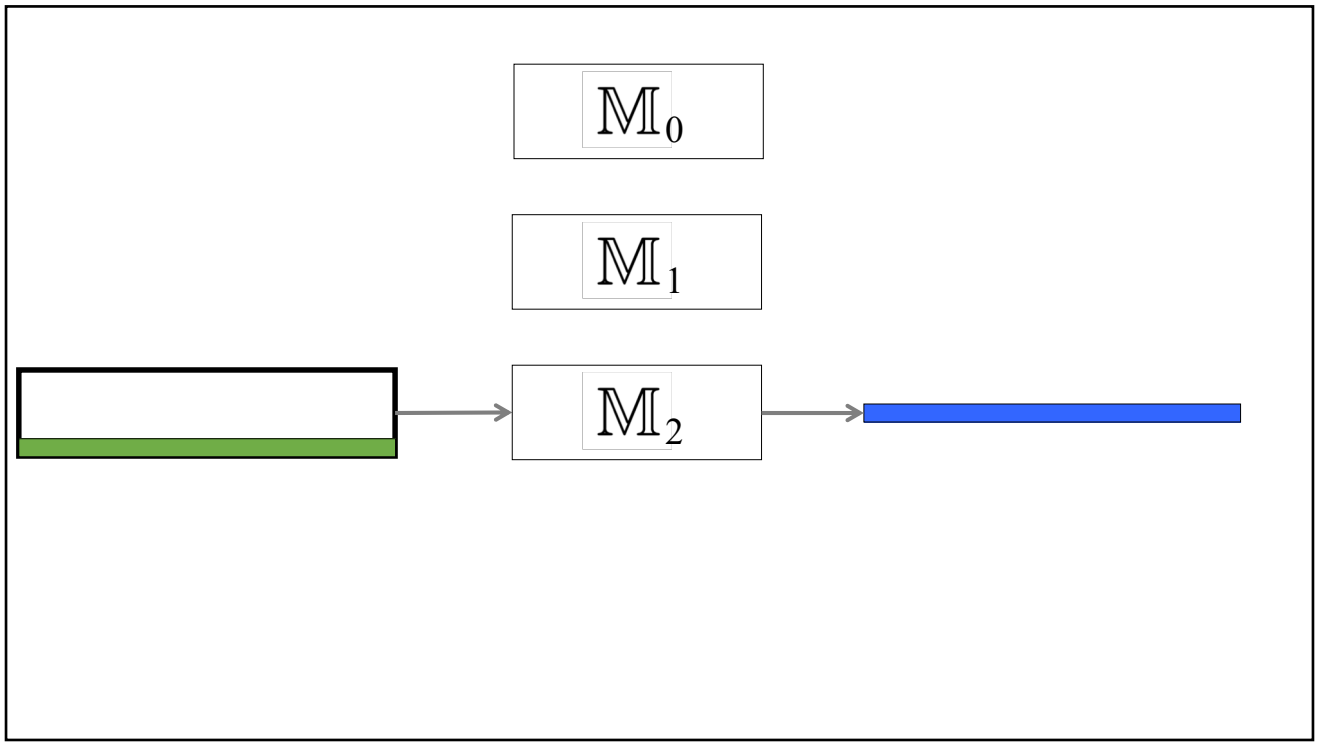
https://en.wikipedia.org/wiki/Gradient_descent#/media/File:Gradient_descent.svg

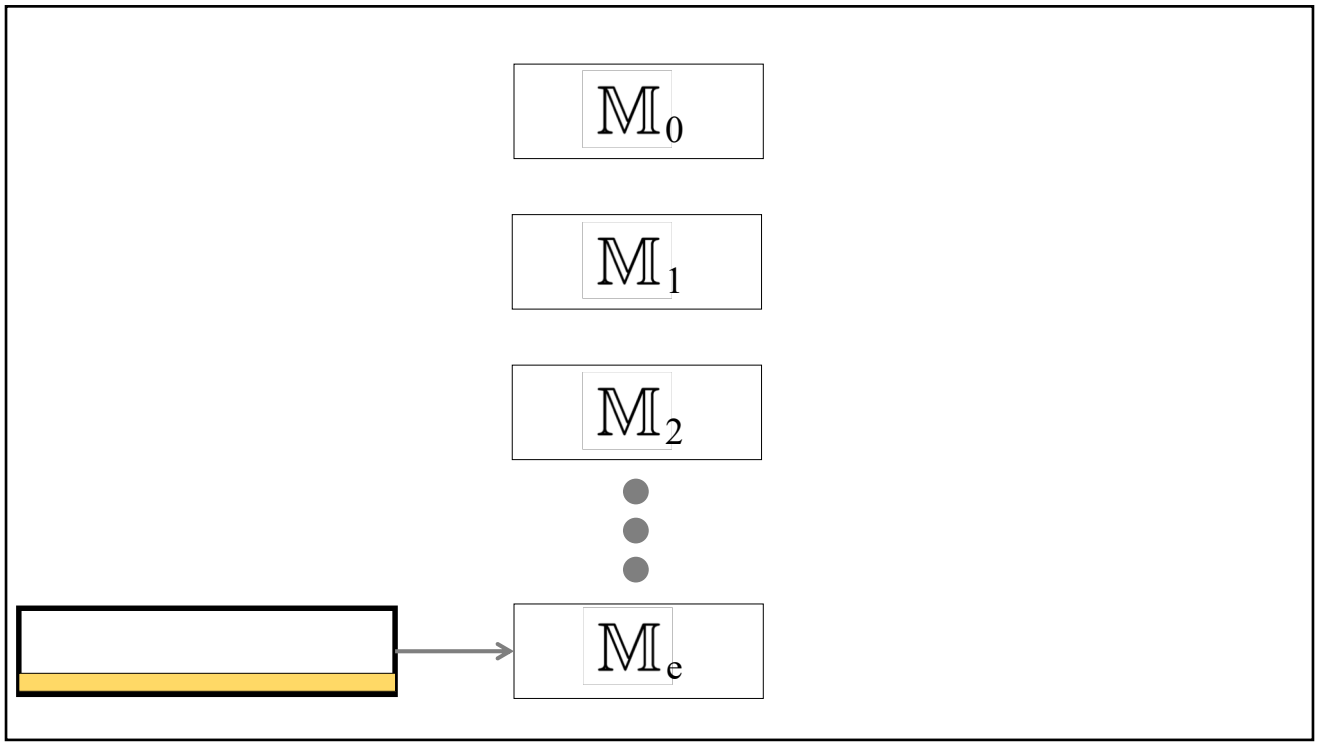

$\mathbb{M}_0$

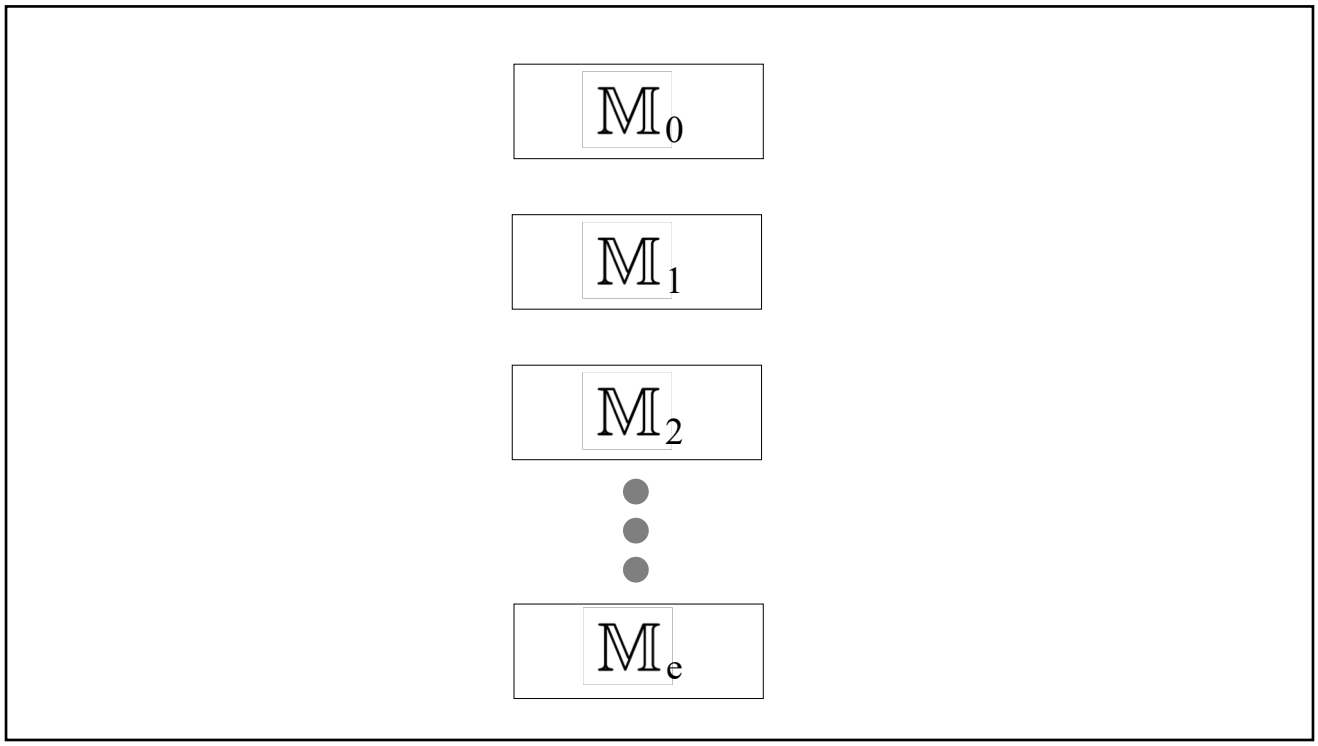

$\mathbb{M}_0$

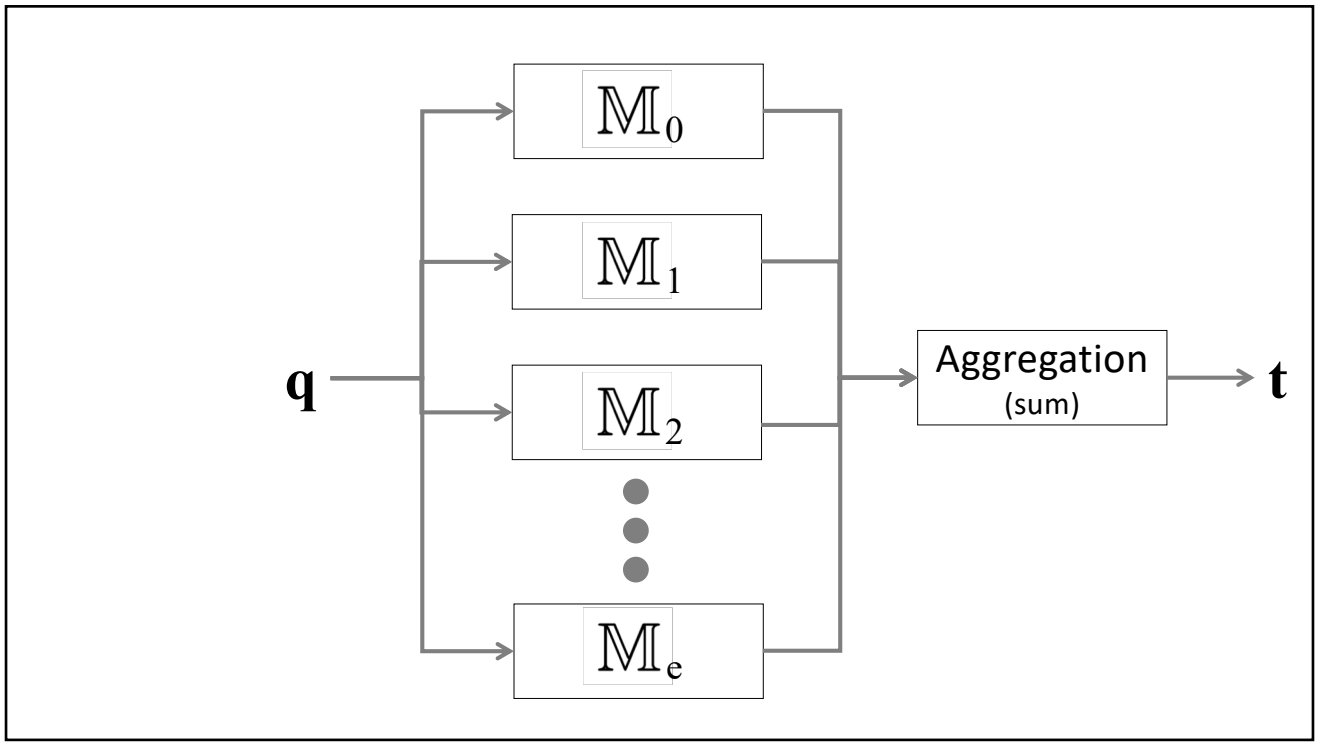

$\mathbb{M}_0$


$\mathbb{M}_0$
$\mathbb{M}_1$


$\mathbb{M}_0$
$\mathbb{M}_1$
$\mathbb{M}_2$


$\mathbb{M}_0$
$\mathbb{M}_1$
$\mathbb{M}_2$
$\mathbb{M}_e$


$\mathbb{M}_0$
$\mathbb{M}_1$
$\mathbb{M}_2$
$\mathbb{M}_e$


q
$\mathbb{M}_0$
$\mathbb{M}_1$
$\mathbb{M}_2$
$\mathbb{M}_e$
Aggregation
(sum)
t

Gradient Boosting Variants

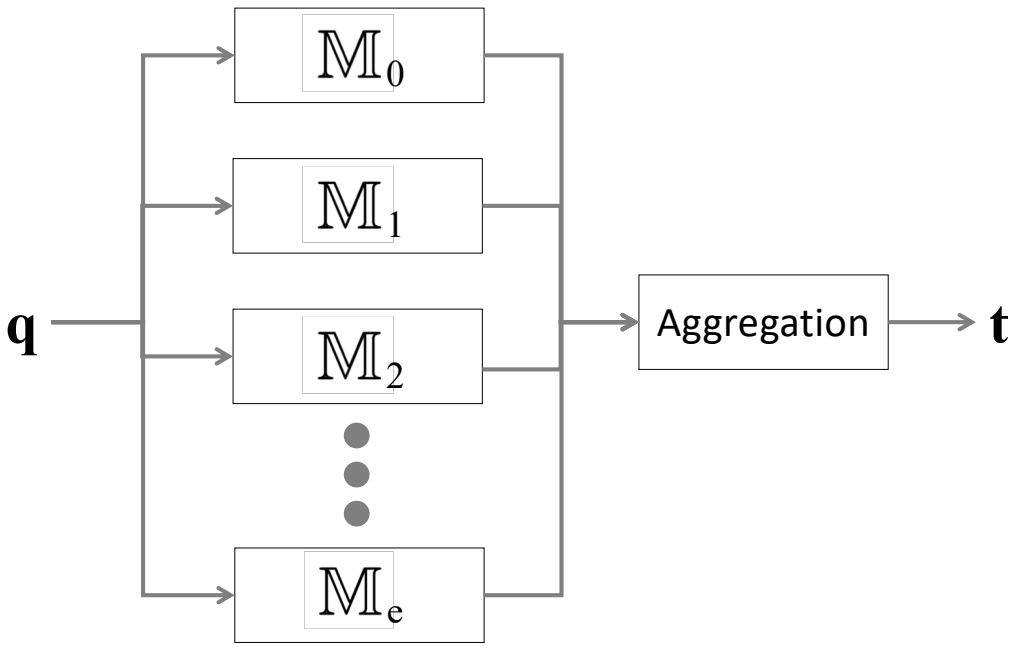
There are lots of variants of gradient boosting

- Different kinds of loss functions are common (least squares, huber, ...)
- Gradient boosting can be implemented with any kinds of base models (small decision trees, ~5 levels, are common)
- Stochastic gradient boosting adds subsampling to each iteration and has been shown to prevent overfitting

Gradient Boosting Variants

- Learning rate is often added which decreases the influence of each subequent tree in a model
- Modifying the algorithm for classification is not difficult - changes in loss functions
- XGBoost is a nice, powerful, scalable implementation oif gradient boosting that is in widespread use

SUMMARY



Summary

Boosting in a general approach to building ensembles.

The AdaBoost algorithm is a specific approach to boosting for binary classification tasks.

Generally works well, but is sentitive to noise in target labels.

Stacking is an interesting approach to building ensembles but is not very widely used

Summary

Gradient boosting is a small modification of the boosting approach, but has been shown to be very effective in practice.

*It used to be **random forest** that was the big winner, but over the last six months a new algorithm called **XGboost** has cropped up, and it's winning practically every competition in the structured data category.*

- Anthony Goldbloom, CEO Kaggle

Anthony Goldbloom gives you the secret to winning Kaggle competitions https://www.import.io/post/how-to-win-a-kaggle-competition/

Questions