Machine Learning for Spaced Repetition in Human Learning

Bianchi Alessandro, De Becker Sophie, Vasquez Emanuel
MAGMA Learning Lab, Supervisor: Maxime Gabella
CS-433 Machine Learning, EPFL, Switzerland

***Abstract*—In this report, we analyse how an exponential human learning model can be used to predict the learning rate of students using the Ari9000 application. In particular, we apply gradient descent and online gradient descent to fit the exponential model. We compare a least squares loss, a regularized loss, and a maximum likelihood estimation. Overall, simple baseline models performed slightly better in terms of prediction accuracy than the exponential model. This later model is limited by how well the features can describe a student's surrounding given available data. Even if our engineered features failed to produce high predictive accuracy, the resulting half-life values could still be useful for creating improved learning schemes.**

I. Background Theory and Problem Formulation

Psychological experiments show that the probability of recall following exposure to a topic, decreases exponentially with time [1]. This behaviour has been described by a half-life model [2]:

$$\hat{p} = 2^{\frac{-\Delta}{\hat{h}}} \tag{1}$$

Where p represents the probability of recall, and h represents the half-life or learning rate. The way specific data features affect the half-life can be modeled through the following exponential:

$$\hat{h} = 2^{\vec{\mathbf{w}} \bullet \vec{\mathbf{x}}} \tag{2}$$

This modeling setup has previously been expanded upon [3] with either new features that take into account topic complexity, or by combining with neural networks.

For this project, we look at how gradient descent can be used to fit this exponential recall model to data from students using Ari9000. Ari9000 is a platform for personalized tutoring that uses artificial intelligence to help students learn more efficiently. Specified questions are presented to students based on their individual knowledge level and memory abilities. Ari9000 also allows for the visualization of learned concepts.

Since length of study intervals have been shown to alter long term knowledge retention [1], obtaining a student's individual learning rate value can be used to adjust the timing of questions and ultimately maximize learning. However, the specific loss function used for gradient descent will depend on the type of data available.

II. Models and Methods

A. Data Outline

Our data consists of timestamped answers from 903 students, as well as the associated topic. Because the exponential model requires the time since last exposure, the time between two answers on the same topic is computed. All topics/questions only viewed once by each student are discarded. Additional features, such as number of previous correct or incorrect answers, are generated. The selection and generation of features to be tested is explained in section IV. From the data we obtain; p: Whether the student was correct [0 or 1]. $\hat{p}$: Predicted probability of correct [0 to 1]. $\hat{h}$: Predicted half-life. Δ: Lag time between same topic questions. $\vec{\mathbf{w}}$: Parameter vector. $\vec{\mathbf{x}}$: Feature vector for a data point.

The following table is an example of data from a single student with a single feature (Total past correct):

TABLE I: Example student data: 3 datapoints with a single feature.

Index	Answer p	Lag Time Δ	Total past correct x
0	1	1	1
1	0	6	2
2	1	40	2

Models are fitted per student. The obtained parameters $\vec{\mathbf{w}}$ can be used to predict the probability of a students answering incorrect (0) or correct (1).

B. Loss function

Three different loss functions were compared. The outline of each function is in Table II and compared to the model used in [2]. Details as to why these loss functions were implemented are explained in section III. The likelihoods for each loss are the following:

- Simple Likelihood:

$$(p - \hat{p})^2 \tag{3}$$

- Regularized Likelihood (Reg):

$$(p - \hat{p})^2 + \frac{\lambda}{2}(D)^2 \tag{4}$$

- Maximum Likelihood Estimation (MLE):

$$-(p \cdot log(\hat{p}) + (1 - p) \cdot log(1 - \hat{p})) \tag{5}$$

C. Data Fitting

Optimization was performed using either gradient descent, or online gradient descent.

Each student submits a set of answers in order. To train our models, the train and test sets are divided chronologically. For a student with N questions, questions 1 through i belong to the train set, and questions $i+1$ through N belong to the test set.

In Gradient Descent (GD), the entire train set is used. Parameters $\vec{\mathbf{w}}$ are initialized at zero, and updated by a factor α based on the gradient of the Loss at the initial $\vec{\mathbf{w}}$ with respect to all points in the test set. This is repeated until the loss no longer decreases, or a maximum number of iterations is reached. This method is the same one used by [2].

TABLE II: Overview of models

	Simple model	**Regularized model**	**MLE model**	**Model by Settles at al.**
Input data	$\vec{p}$: answers correct [1] or incorrect [0] $\vec{\Delta}$: lag times between questions $\mathbf{X}$: feature matrix of size N datapoints $\times$ M Features			$\vec{p}$: recall rate ranging [0,1] $\vec{\Delta}$: lag time between questions $\mathbf{X}$: feature matrix of sizeß N datapoints $\times$ M Features
Common Variables	$\hat{p}$: Predicted probability of answering correctly $\hat{h}$: Predicted half-life $\vec{\mathbf{w}}$: Parameter vector of size $1 \times M$ Features $\vec{\mathbf{x}}$: Data-point vector of size $1 \times M$ Features $\vec{\mathbf{w}} \bullet \vec{\mathbf{x}} = w_1 \cdot x_1 + w_2 \cdot x_2 + ... + w_{M-1} \cdot x_{M-1} + w_M \cdot x_M$			
Unique Variables		$D = f(\vec{\mathbf{w}}, p, \vec{\Delta}, \vec{\mathbf{x}})$: Smallest distance between $\vec{\mathbf{w}}$ and hyperplane bisecting l λ: Scaling factor		h_e: Estimated half-life α: h Scaling factor λ: Scaling factor
Loss Function	$L(\langle \vec{p}, \vec{\Delta}, \mathbf{X} \rangle; \vec{\mathbf{w}}) = \frac{1}{N} \sum_{i=1}^{N} l(\langle p, \Delta, \vec{\mathbf{x}} \rangle_i; \vec{\mathbf{w}})$			
$l(\langle p, \Delta, \vec{\mathbf{x}} \rangle_i; \vec{\mathbf{w}}) =$	$(p - \hat{p})^2$	$(p - \hat{p})^2 + \frac{\lambda}{2}(D)^2$	$-(p \cdot log(\hat{p}) + (1-p) \cdot log(1-\hat{p}))$	$(p - \hat{p})^2 + \alpha(h_e - \hat{h})^2 + \lambda \|\vec{\mathbf{w}}\|_2^2$
Where	$\hat{p} = 2^{\frac{-\Delta}{\hat{h}}}$ $\hat{h} = 2^{\vec{\mathbf{w}} \bullet \vec{\mathbf{x}}}$	$\hat{p} = 2^{\frac{-\Delta}{\hat{h}}}$ $\hat{h} = 2^{\vec{\mathbf{w}} \bullet \vec{\mathbf{x}}}$ $D = \frac{\vec{\mathbf{x}}}{\|\vec{\mathbf{x}}\|} \vec{\mathbf{w}} + \frac{I}{\|\vec{\mathbf{x}}\|}$ $I = -log_2(\frac{-\Delta}{log_2(\|p - \sqrt{0.001}\|)})$	$\hat{p} = 2^{\frac{-\Delta}{\hat{h}}}$ $\hat{h} = 2^{\vec{\mathbf{w}} \bullet \vec{\mathbf{x}}}$	$\hat{p} = 2^{\frac{-\Delta}{\hat{h}}}$ $\hat{h} = 2^{\vec{\mathbf{w}} \bullet \vec{\mathbf{x}}}$ $h_e = \frac{-\Delta}{log_2(p)}$
Loss Gradient	$\nabla_{\vec{\mathbf{w}}} L(\langle \vec{p}, \vec{\Delta}, \mathbf{X} \rangle; \vec{\mathbf{w}}) = \frac{1}{N} \sum_{i=1}^{N} \nabla_{\vec{\mathbf{w}}} l(\langle p, \Delta, \vec{\mathbf{x}} \rangle_i; \vec{\mathbf{w}})$			
$\nabla_{\vec{\mathbf{w}}} l(\langle p, \Delta, \vec{\mathbf{x}} \rangle_i; \vec{\mathbf{w}}) =$	$-2 \cdot (p - \hat{p}) \cdot \nabla_{\vec{\mathbf{w}}} \hat{p}$	$-2 \cdot (p - \hat{p}) \cdot \nabla_{\vec{\mathbf{w}}} \hat{p}$ $+ \lambda \cdot \frac{\vec{\mathbf{x}}}{\|\vec{\mathbf{x}}\|} \cdot D$	$-\nabla_{\vec{\mathbf{w}}} \hat{p} \cdot \frac{p}{\hat{p}}$ $+ \nabla_{\vec{\mathbf{w}}} \hat{p} \cdot \frac{1-p}{1-\hat{p}}$	$-2 \cdot (p - \hat{p}) \cdot \nabla_{\vec{\mathbf{w}}} \hat{p}$ $+ 2\alpha(\hat{h} + \frac{\Delta}{log_2(p)}) \cdot ln(2) \cdot \hat{h} \cdot \vec{\mathbf{x}}$ $+ 2\lambda \vec{\mathbf{w}}$
Where	$\nabla_{\vec{\mathbf{w}}} \hat{p} = \vec{\mathbf{x}} \cdot \Delta \cdot ln^2(2) \cdot \frac{1}{\hat{h}} \cdot \hat{p}$			

In Online Gradient Descent (Online), only one point is used at a time, when that point is added to the train set. The parameters $\vec{\mathbf{w}}$ are only updated based on the loss gradient with respect to that point. This method is equivalent to a chronologically ordered Stochastic Gradient Descent.

D. Model Assessment

The objective is to obtain parameters $\vec{\mathbf{w}}$ that are representative of a student's learning rate in as few answers as possible. To show how quickly a model is adapting, we take increasingly large training sets, fit, predict and compute accuracies.

As an example, for a student with N answers: we start with a train set of size 1 and test set of size $N-1$ and parameters $\vec{\mathbf{w}} = 0$. We fit, obtain parameters, predict the test set, and obtain an accuracy metric. We then repeat with a train set of size 2 and test set of size $N-2$ and initialize with the parameters obtained from the previous train set. This is repeated until we reach a train set of size $N-1$. This leaves us with series of parameter sets, train/test losses, and train/test accuracies of size $N-1$; one point for each train set size.

Accuracy can be computed from the test set by:

A) Predicting the immediate next answer: The output is a binary (predicted correct or incorrect). To obtain an accuracy value, from this, the values must be averaged out over multiple students.

B) Predicting all the remaining answers: The output is an accuracy estimate whose reliability is proportional to the size of the test set (remaining answers).

The accuracy series can be synthesized by:

1) First finding average performance of each student s, and then averaging across T students.

2) First finding average performance for each answer i seen across all students, then average across N answers. Because each student has a different number of answers, at a given i students will have train set of same sizes, but tests sets of different sizes. Depending on the metric and averaging method this might change the interpretation.

For notation purposes the following are used:
tsa: averaging method 2 + metric A
$tsa1$: averaging method 2 + metric B
$tsa2$: averaging method 1 + metric A
$tsa3$: averaging method 1 + metric B

As an overall metric, we multiplied the average accuracy across the first 50 questions, by the average accuracy for all the remaining questions. The first average is large if the method reaches high accuracies quickly, and the second is large if the method has overall good accuracy. We called this metric ts_{perf}:

$$ts_{perf} = \frac{1}{50} \sum_{i=1}^{50} (\frac{1}{T} \sum_{s=1}^{T} acc_{i,s}) \cdot \frac{1}{N-50} \sum_{i=51}^{N} (\frac{1}{T} \sum_{s=1}^{T} acc_{i,s}) \tag{6}$$

Where $acc_{i,s}$ is the accuracy measure **B** at train set size i, for student s. This method was applied for student with at least 51 answer. During hyperparameter tuning we also only focused on students with less than 300 answers to limit computing time.

E. Baseline models

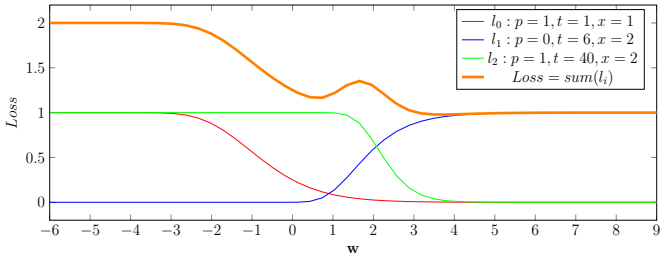
For each student, we used the best of two baseline models to compare our model with: 1) Predicting only the majority outcome (1 if the student answers mostly correct, or 0 if the student answers mostly incorrect). 2) Predicting the same outcome as the previous question.

III. Regularization of Loss Function

We initially used our simple likelihood Equation3, for our optimizations. However, we noticed that the results were dependent on initial parameters. Unlike in the study [2], since we use true recall values of only 0 or 1, the likelihood will be a monotonic, sigmoid-like curve. As such, the overall Loss

function is not guaranteed to have a definite global minimum. This is illustrated in Figure 1 with our example student from Table I. At extreme $\mathbf{w}$, ∇L is close to zero, making gradient descent inefficient at finding local minima. In our testing, gradient descent would not update $\vec{\mathbf{w}}$.

Fig. 1: Example Loss Curve for $l_i = (p - \hat{p}(\mathbf{w}))^2$



A. Regularization

The first solution was to include regularization in the loss function. Because we wanted our parameters to be Student specific, this regularization had to be data dependent. We decided to use the distance of the parameters $\vec{\mathbf{w}}]$ from the bottom corner of the sigmoid $l(\vec{\mathbf{w}})$ as a regularization value. The "bottom corner", was mathematically represented as a hyper-plane that orthogonally bisects the sigmoid at $l = 0.001$. This type of regularization is dependent on each data-point, and as such, student specific. It basically allows gradient descent to find at least a local minimum.

B. Maximum Likelihood Approach

Another solution is minimizing an alternative loss function. Inspired by logistic regression, the optimization problem is reformulated as maximizing the likelihood (MLE formulation). Except now the probability isn't modelled by the sigmoid function: $\sigma(x) = \frac{1}{1+e^{-\vec{\mathbf{w}} \bullet \vec{\mathbf{x}}}}$ but our half life regression probability model: $\hat{p}(\langle \Delta, \vec{\mathbf{x}} \rangle; \vec{\mathbf{w}}) = 2^{\frac{-\Delta}{2^{-\vec{\mathbf{w}} \bullet \vec{\mathbf{x}}}}}$.

Further details for both Regularization and MLE are in the Appendix.

IV. Feature Engineering

In order to improve the accuracy of our model, we engineered a number of additional features, that take into account question and student-specific data. For this purpose, an additional dataset, which maps topics to higher level concepts, was integrated with the given one.

A. Feature Generation

The features are described in Table III.

V. Implementation and Results

A. Baseline Models

		Metrics			
		tsa	tsa1	tsa2	tsa3
Previous Baseline Model (PBL)	id: 1572	72.0%	74.9%	N/A	N/A
	id:2470	65.2%	65.0%	N/A	N/A
	us val	74.7%	56.8%	79.9%	79.3%
	us test	73.0%	52.1%	70.7%	74.8%
Majority Baseline Model (MBL)	id: 1572	84.8%	85.2%	N/A	N/A
	id:2470	73.4%	74.0%	N/A	N/A
	us val	82.9%	66.3%	86.7%	86.1%
	us test	82.9%	63.2%	81.7%	83.2%

TABLE IV: 2 Baseline model results for the different metrics. Two individual user performances are reported as well as the average performances for the validation and test set which correspond to the same split as the one used in hyperparameter tuning

B. Hyper-parameters

Due to time length of computing time, for each model, a range of hyper-parameters was tested for on a subset of 70 students (50 tuning/validation, 20 for testing). The optimal hyper-parameters were then used to compute the accuracy on the entire student set.The hyper-parameters investigated were the following: Gradient Descent: learning rate, number of features considered, maximum iterations
Regularized Gradient Descent: learning rate, number of features considered, lambda
Online Gradient Descent: learning rate, number of features considered
MLE: learning rate, number of features considered, maximum iterations

Following the hyper-parameter tuning procedure, the Regularized Gradient Descent proved to yielded the best performance metric (ts_{perf}).

Measure	us val	us test
tsa	75.4%	78.5%
tsa1	76.9%	80.7%
tsa2	72.9%	73.7%
tsa3	76.9%	76.2%
ts perf	0.6095	0.5903

TABLE V: Train and test results obtained using Regularized Gradient Descent with the following fine-tuned parameters: features used=[5,6,7,8,11,12,13,14,19,20], gamma=1, lambda=0.001, it=1

C. Feature Exploration

2 students were selected at random (id: 2470 and 1572). For these two students GD with gamma 0.1 and max_iter 10 was implemented as well as online GD. It would take too long to carry this out for all students. For completeness ideally we would also vary the hyperparameters but here they stayed fixed. First we iterate over each feature and consider a 1 variable model and order the features by descending performance measure (ts_{perf}) in a list. We then repeat the two models but this time changing the number of features considered by increasing the slice size of the ordered features list. The results shown indicated that feature of index 5 (Table III) is important in all scenarios. With this information we can confirm including number of total correct even if it isn't topic dependent can add value. However a noteworthy observation is that for id:1572 all features performed the same and increasing feature number considered didn't seem to help, so in some cases it seems like we don't have all the features necessary to model the scenario well.

TABLE III: Features

Feature Name	Feature Description	Feature Index	Feature Explanation
previous_correct	# past correct answers	5	insight on student's past performance
previous_incorrect	# past incorrect answers	6	insight on student's past performance
previous_correct_topic	# past correct answers for given topic	7	insight on student's past performance on the topic
previous_incorrect_topic	# past incorrect answers for given topic	8	insight on student's past performance on the topic
n_topics_since_last_revision	# unique topics seen since the last revision of the given topic	11	insight on student's activities between two revisions: seeing many different topics may negatively impact recall
n_questions_since_last_revision	# questions answered since the last revision of the given topic	12	insight on student's activities between two revisions: seeing many questions may negatively impact recall
n_similar_topics_since_last_revision	# unique topics seen since the last revision that are similar to the given one. Similar topics are topics that have one or more macro topic in common	13	insight on student's activities between two revisions: seeing related topics may positively impact recall
last_revision_correct	binary feature that specifies whether the last question answered for the same topic was correct or not	14	insight on student's performance in the most recent time the topic was seen as it may be more impactful than the entire history
n_clicks	# clicks needed to answer the question	19	insight on the type of question: a question with multiple gaps to fill may be harder
n_topics	# higher level topics that the lower level topic is mapped to	20	insight on the macro topics the question is about: a question spanning over many macro topics may be more complicated to answer

D. Robustness

Te measure robustness: for 10 students, the answers from each student were randomly re-ordered, and split into 70%/30% train/test set. Each model (Reg, Online, and MLE) were fit and the parameters $\vec{\mathbf{w}}$ obtained. This was repeated 100 times, giving us 100 parameters set for each student and method. This allows us to compute standard deviations for parameters, when a method is given the same data-set just presented in a different order. The standard deviations in $\vec{\mathbf{w}}$ were: Reg (0.309), Online (0.752), and MLE (0.485) given an average $\vec{\mathbf{w}}$ on the scale of 0.966. As such, the standard deviation is quite large compared to the mean value, however online has a much larger deviation.

E. Results

In terms of accuracy: Regularized works on average marginally better, but all reach similar accuracies. Depending on how the metric is computed, best average accuracies ranged between 76% and 91%. In terms of computation speed: Regularized online is faster, even when compared to 1-step gradient descent optimizations for both Regularized and MLE. However, in terms of robustness: Online performed worst, and Regularized performed best.

VI. Discussion

The instances that the model failed to predict, often were ones that could not fit in the exponential model: wrong answers with very small time delays, or correct answers with very large time delays. These instances can be explained by the features not capturing the underlying mechanism. For example, if a student views an answer outside of the application, this could cause a correct reply even after a long time gap. Conversely if a student is not paying attention, they may give an incorrect answer to the same question they had recently viewed. Our feature generation was limited by what data we had on the student.

Due to computational time, not all hyper-parameters combination were tested. Finer tuning could be carried out by varying additional hyper-parameters; this could include choosing different bases for the exponential formulation, different learning rates, larger number go gradient descent iterations, applying polynomial expansion to the features, or testing all combinations of our current 8 features. With this, higher prediction accuracy could be reach. However, the largest increase is likely to come from feature engineering.

The original psychological papers [1] that looked at the exponential decrease memory were performed under very controlled conditions. Ultimately, how well we can fit an exponential model to the data will depend on how well the features we engineered take into account the student's environment.

VII. Conclusion

Given the state of the art in machine learning, a different type if model might be better at predicting a student's answer. In fact, even a simple model, predicting the same answer as before, often outperformed our fitted exponential models. However, this project's goal was to obtain a tangible metric from the student: recall half-life. This metric can be used to help a student learn, by scheduling their study questions accordingly. While our models did not predict all the students answers perfectly, the underlying half-life values may be sufficiently precise for practical use.

References

[1] N. J. Cepeda, E. Vul, D. Rohrer, J. T. Wixted, and H. Pashler, "Spacing effects in learning: A temporal ridgeline of optimal retention," *Psychological science*, vol. 19, no. 11, pp. 1095–1102, 2008.

[2] B. Settles and B. Meeder, "A trainable spaced repetition model for language learning," in *Proceedings of the 54th annual meeting of the association for computational linguistics (volume 1: long papers)*, pp. 1848–1858, 2016.

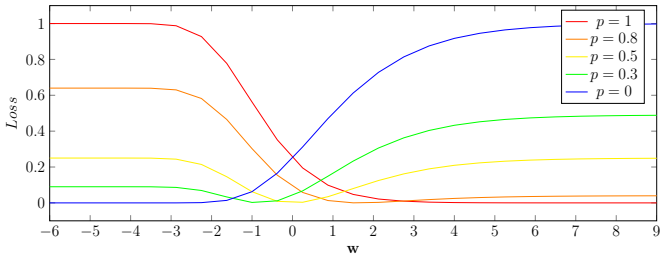
[3] A. Zaidi, A. Caines, R. Moore, P. Buttery, and A. Rice, "Adaptive forgetting curves for spaced repetition language learning," *arXiv preprint arXiv:2004.11327*, 2020.

APPENDIX

A. Illustration of regularization

If we plot simple likelihood for different true recall values p, we obtain the following plot:

Fig. 2: Effect of True Recall Value on Likelihood Curve $l_i = (p - \hat{p}(\mathbf{w}))^2$



The **Dist** function works as follows: for each point, we calculate a hyper-plane that orthogonaly bisects the sigmoid curve at a specific value. We chose bisection at $l_i = 0.001$ since we want a small loss and a centered $\mathbf{w}$). We then return the smallest distance of $\mathbf{w}$ from the hyper-plane. This distance squared is added to l_i resulting in the loss function:

$$(p - \hat{p})^2 + \frac{\lambda}{2}(Dist(\tilde{\mathbf{w}}, p, \Delta, \tilde{\mathbf{x}}))^2 \tag{7}$$

This concept is illustrated in 1 dimension on Figures 3 and 4. However, this 1D example can be extended to any dimension by thinking of the bisecting line as hyper-plane that summarizes to location and orientation of the sigmoid shaped l_i.

Fig. 3: Regularized Loss Curve for $l_i^* = l_i + 0.5(Dist_i(\mathbf{w}))^2$

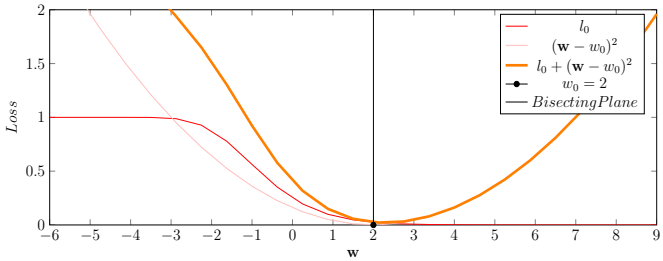
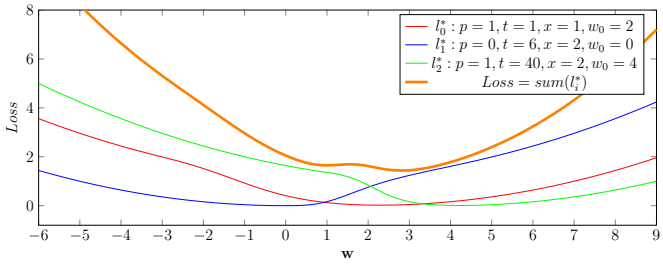


Fig. 4: Example Regularized Loss Curve



Our example student, Figure 4 illustrates how the regularized loss function no longer has a plateau for $\mathbf{w} < -4$ and $\mathbf{w} > 4$. This guarantees that at least some local minima is found.

B. Derivation of gradient for the MLE method

the probability for one data point:

$$P(Y = y|X = x) = f_w(x)^y(1 - f_w(x)^1 - y$$

where $f_w(x) = 2^{\frac{-t}{h_w(x)}}$ and $h_w(x) = 2^{w^T x}$

given n time observations for a given student the likelihood is expressed as follows

$$\prod_{i=1}^{n} f_w(x_i)_i^y(1 - f_w(x_i)^{1-y_i}$$

The aim is to maximize the MLE, or equivalently minimize (-MLE). Similarly to logistic regression we will minimize the -log(MLE).

$$L(w) = -\sum_{i=1}^{n} y_i log(f_w(x_i)) + (1 - y_i) log(1 - f_w(x_i))$$

The gradient of this complex formulation is derived below using the chain rule:

$$\frac{\partial L(w)}{\partial w_j} = -\sum_{i=1}^{n} y_i \frac{1}{f_w(x_j} \cdot \frac{\partial f_w(x_i)}{\partial w_j}) + (1 - y_i) \frac{1}{1 - f_w(x_i} \frac{\partial f_w(x_i)}{\partial w_j})(-1)$$

$$\frac{\partial f_w(x_i)}{\partial w_j} = ln(2)^2 f_w(x_i) \frac{1}{h_w(x_i)} t x_i^j$$

x_i^j is the j^{th} dimension of i^{th} observation x_i.

To be more precise, if we only look at i'th observation's contribution to j'th component of gradient it is $\frac{\partial L(w)}{\partial w_j}_i = -y_i \frac{1}{f_w(x_j)} \cdot \frac{\partial f_w(x_i)}{\partial w_j}) + (1 - y_i) \frac{1}{1 - f_w(x_i)} \frac{\partial f_w(x_i)}{\partial w_j})$ When implementing gradient descent the gradient was also normalized by the number of observations used to compute it