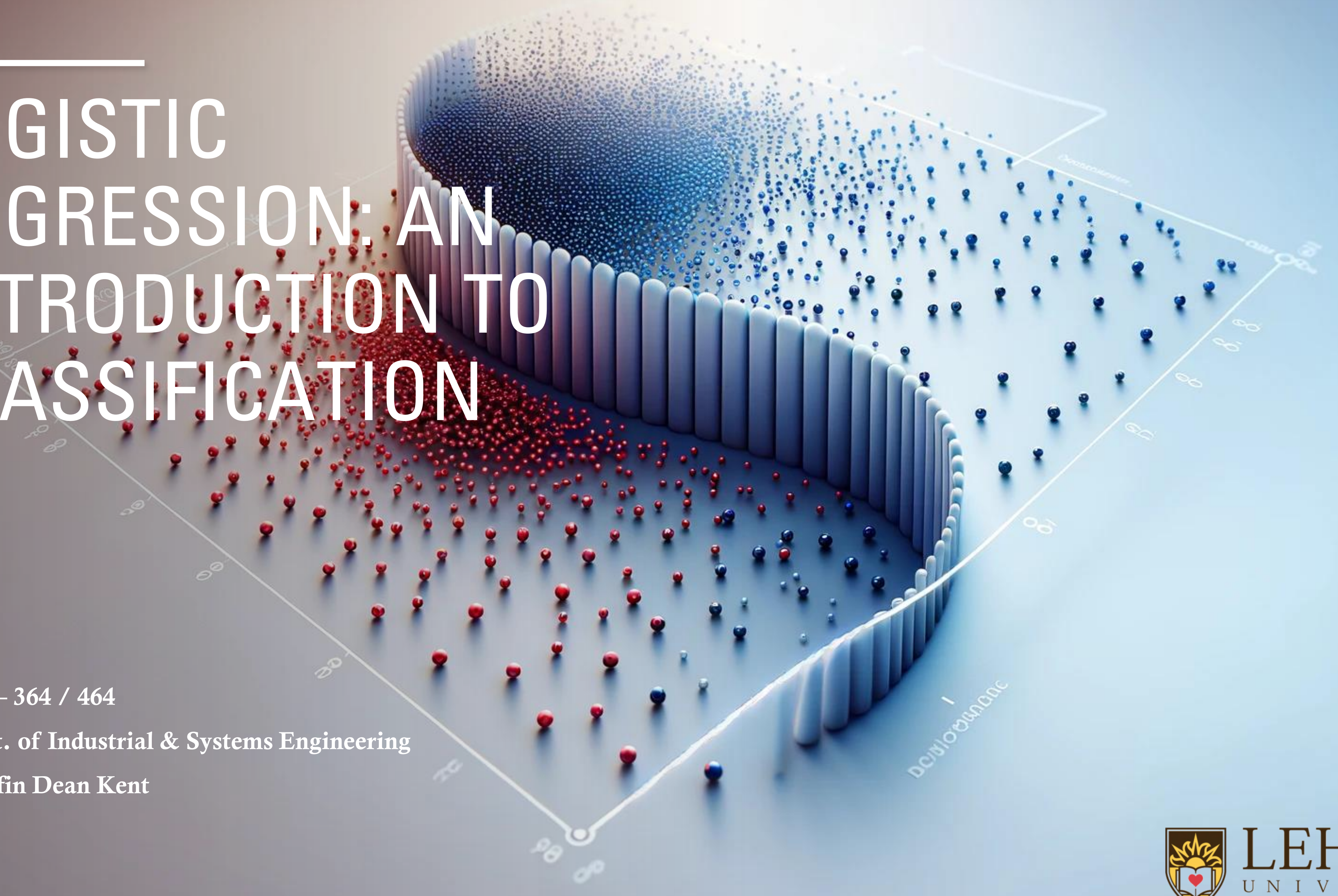
LOGISTIC REGRESSION: AN INTRODUCTION TO CLASSIFICATION



- ISE – 364 / 464
- Dept. of Industrial & Systems Engineering
- Griffin Dean Kent

LEHIGH UNIVERSITY

LOGISTIC REGRESSION MODEL

Why Logistic Regression?

Instead of predicting some real number $y \in \mathbb{R}$, what if we were trying to predict a binary target $y \in \{0,1\}$?

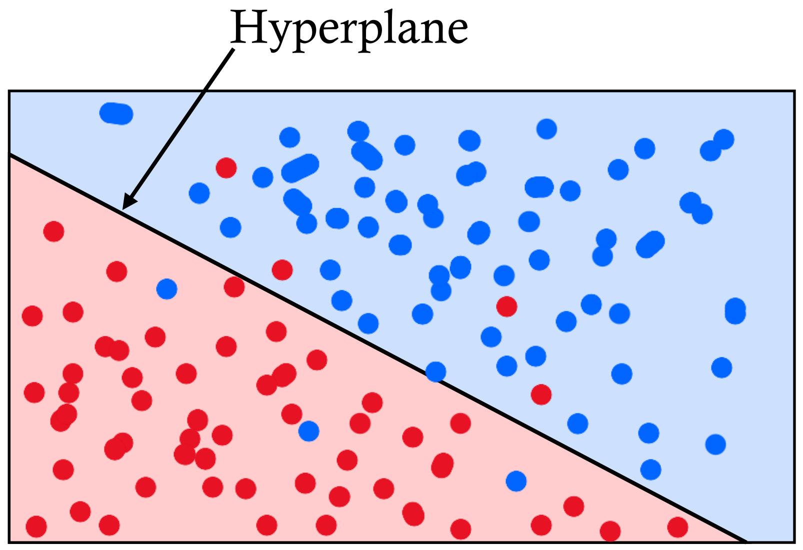
We utilize a ***Logistic Regression Model*** $h: \mathbb{R}^{n+1} \rightarrow [0,1]$ which is the parameterized function

$$h_\theta(x) = h_\theta(x; \theta) = \sigma(\theta^T x),$$

Where $\sigma: \mathbb{R} \rightarrow [0,1]$ is the ***sigmoid function*** (do not confuse this with standard deviation).

Further, we will denote $z \coloneqq \theta^T x$. Hence, $h_\theta(x) = \sigma(z)$.

INTUITION BEHIND LOGISTIC REGRESSION



As stated, a logistic regression model is utilized to predict a binary target variable, i.e., **class A** and **class B**.

What type of mathematical model do you think would be a natural first choice to predict a binary target variable?

Answer: ***Separating hyperplanes***.

Specifically, logistic regression goes about discerning between two target classes by fitting a ***separating hyperplane*** to the dataset in a way that maximizes the number of correctly classified datapoints.

The question of *how* logistic regression goes about fitting this hyperplane is what we will explore.

THE SIGMOID FUNCTION

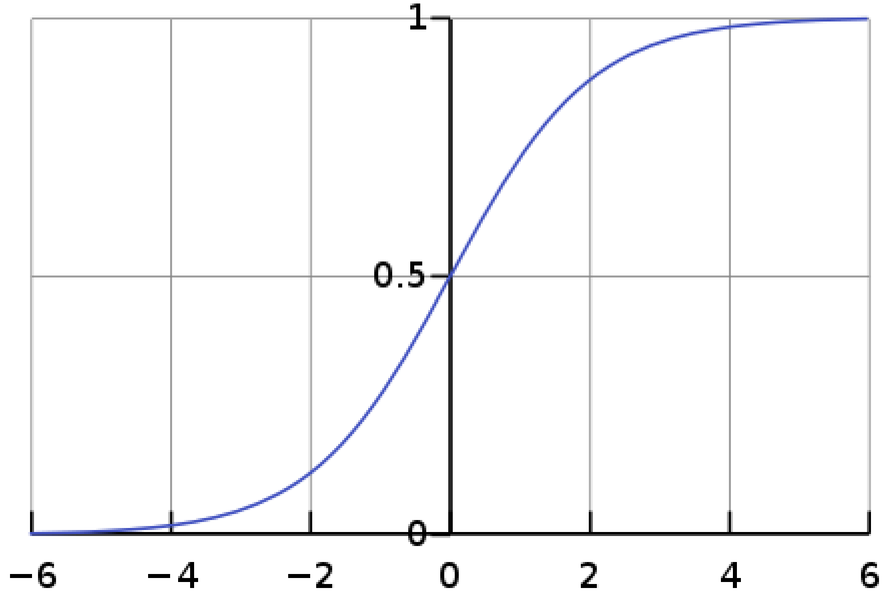
A sigmoid function is a function that has a characteristic "S"-shaped curve.

In mathematics, the term sigmoid is used to refer to a variety of "S"-shaped functions; however, regarding machine learning, we will use the term to refer to the following equation:

$$\sigma(z) \coloneqq \frac{1}{1+e^{-z}}.$$

Do you notice what this function does?

It takes any real number $z \in \mathbb{R}$ and converts it into a ***probability***!

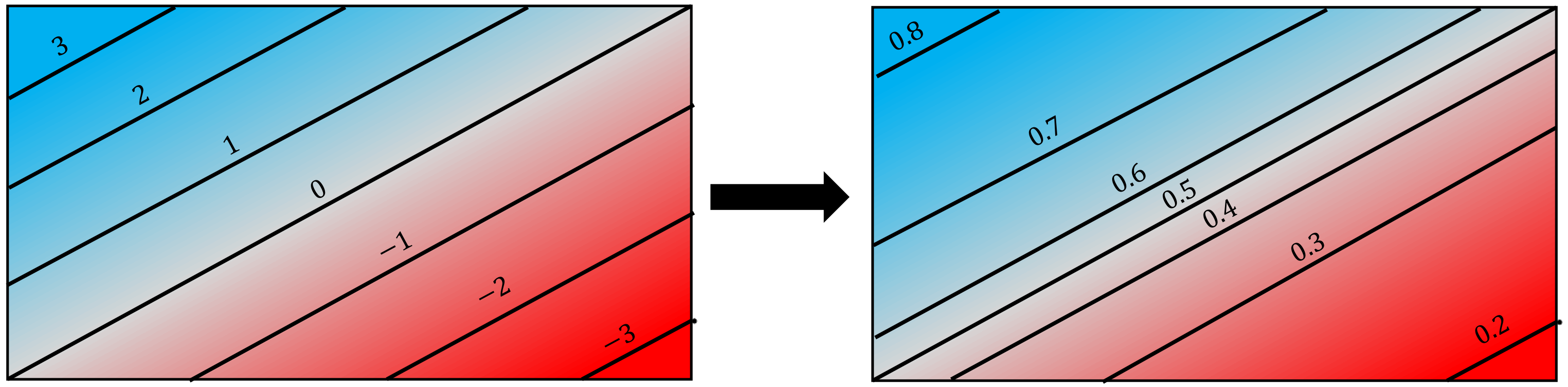


INTUITION BEHIND THE SIGMOID FUNCTION

As stated on the previous slide, the *sigmoid function* is a *mapping* from any real-valued number to a value in the range (0,1), i.e., a *probability*.

Upon further analysis of the sigmoid function, it is immediately clear that all positive values are mapped to probabilities in the range (0.5, 1), all negative values are mapped to probabilities in the range (0, 0.5), and 0 is mapped to a probability of 0.5.

In this way, we can assign datapoints that have a sigmoid value in (0.5, 1) to **class A** and the datapoints that have a sigmoid value in (0, 0.5) to **class B**. Hence, by doing this, there is a natural binary mapping.



PROPERTIES OF THE SIGMOID FUNCTION

- The sigmoid function $\sigma(z)$ is a monotonically increasing function in $z \in \mathbb{R}$.

- $1 - \sigma(z) = \sigma(-z)$.

Proof: Homework.

- $\frac{d}{dz}\sigma(z) = \sigma(z)\sigma(-z)$.

Proof:

$$\frac{d}{dz}\sigma(z) = \frac{d}{dz}(1+e^{-z})^{-1} = -(1+e^{-z})^{-2}\frac{d}{dz}(1+e^{-z}) = (1+e^{-z})^{-2}e^{-z}$$
$$= \left(\frac{1}{1+e^{-z}}\right)\left(\frac{e^{-z}}{1+e^{-z}}\right) = \sigma(z)\big(1-\sigma(z)\big) = \sigma(z)\sigma(-z).$$

LOGISTIC REGRESSION – A PROBABILISTIC INTERPRETATION

Remember, we are dealing with a binary target variable $y \in \{0,1\}$. This means, y will have a **Bernoulli distribution function** (conditioned on x) given by

$$\mathbb{P}\left(y^{(i)} \middle| x^{(i)};\theta\right) = \mathbb{P}\left(y^{(i)} = 1 \middle| x^{(i)};\theta\right)^{y^{(i)}} \cdot \mathbb{P}\left(y^{(i)} = 0 \middle| x^{(i)};\theta\right)^{1-y^{(i)}}.$$

With a **logistic regression** model defined as the function $h_\theta(x) = \sigma(\theta^T x) \in \{0,1\}$, this can be **interpreted** as the <u>probability of observing $y = 1$ conditioned on the input x with parameter θ</u>, i.e.,

$$h_\theta(x) = \sigma(\theta^T x) = \mathbb{P}(y = 1|x;\theta).$$

Under this interpretation, it is clear that $1 - h_\theta(x) = 1 - \mathbb{P}(y = 1|x;\theta) = \mathbb{P}(y = 0|x;\theta)$.

FITTING A LOGISTIC REGRESSION MODEL VIA MAXIMUM LIKELIHOOD

Intuitively, since our logistic regression model is defined as a probability function parameterized by θ, we can go about "fitting" this model via maximizing its likelihood function. The likelihood function is given by

$$L(\theta) = L\left(\theta; \{(x^{(i)}, y^{(i)})\}_{i=1}^{m}\right) = \prod_{i=1}^{m} \mathbb{P}\left((y^{(i)} | x^{(i)}); \theta\right)$$

$$= \prod_{i=1}^{m} \mathbb{P}(Y = 1 | x^{(i)}; \theta)^{y^{(i)}} \mathbb{P}(Y = 0 | x^{(i)}; \theta)^{1-y^{(i)}}$$

$$\prod_{i=1}^{m} \sigma(z^{(i)})^{y^{(i)}} \sigma(-z^{(i)})^{1-y^{(i)}},$$

where $z^{(i)} \coloneqq \theta^T x^{(i)}$. Therefore, **training a logistic regression** model is equivalent to solving the optimization problem

$$\theta^* = \underset{\theta \in \mathbb{R}^{n+1}}{\operatorname{argmax}} L(\theta).$$

THE CROSS-ENTROPY LOSS

Similar to linear regression (in the probabilistic setting), to get rid of the products, we will instead minimize the negative log-likelihood function, where the log-likelihood function is given by

$$\ell(\theta) = \log L(\theta) = \sum_{i=1}^{m} \left[y^{(i)} \log\left(\sigma\left(z^{(i)}\right)\right) + \left(1 - y^{(i)}\right) \log\left(\sigma\left(-z^{(i)}\right)\right)\right].$$

The negative log-likelihood $-\ell(\theta)$ is also referred to as the "**cross-entropy**" loss function.

Thus, logistic regression is the process of solving the optimization problem

$$\theta^* = \underset{\theta \in \mathbb{R}^{n+1}}{\operatorname{argmax}} L(\theta) = \underset{\theta \in \mathbb{R}^{n+1}}{\operatorname{argmin}} -\ell(\theta)$$

DERIVATION OF THE GRADIENT OF ℓ

(FOR EACH INDIVIDUAL j COMPONENT)

$$\boldsymbol{\nabla}\ell(\boldsymbol{\theta}) = \sum_{i=1}^{m}\left[y^{(i)}\frac{\partial}{\partial\theta}\log\left(\sigma\left(z^{(i)}\right)\right) + \left(1 - y^{(i)}\right)\frac{\partial}{\partial\theta}\log\left(\sigma\left(-z^{(i)}\right)\right)\right]$$

$$= \sum_{i=1}^{m}\left[y^{(i)}\sigma\left(z^{(i)}\right)^{-1}\frac{\partial}{\partial\theta}\sigma\left(z^{(i)}\right) + \left(1 - y^{(i)}\right)\sigma\left(-z^{(i)}\right)^{-1}\frac{\partial}{\partial\theta}\left(1 - \sigma\left(z^{(i)}\right)\right)\right]$$

$$= \sum_{i=1}^{m}\left[y^{(i)}\sigma\left(-z^{(i)}\right)x^{(i)} - \left(1 - y^{(i)}\right)\sigma\left(z^{(i)}\right)x^{(i)}\right]$$

$$= \sum_{i=1}^{m}\left[y^{(i)} - \sigma\left(z^{(i)}\right)\right]x^{(i)} = \sum_{\boldsymbol{i=1}}^{\boldsymbol{m}}\left[\boldsymbol{y}^{(\boldsymbol{i})} - \boldsymbol{h_\theta}\left(\boldsymbol{x}^{(\boldsymbol{i})}\right)\right]\boldsymbol{x}^{(\boldsymbol{i})}$$

$$= \boldsymbol{X}^{\boldsymbol{T}}[\boldsymbol{y} - \boldsymbol{h_\theta}(\boldsymbol{x})],$$

where $h_\theta(x) \coloneqq \left[\sigma\left(z^{(1)}\right), \sigma\left(z^{(2)}\right), \dots, \sigma\left(z^{(m)}\right)\right]^T$.

PROPERTIES OF THE SOLUTION FOR LOGISTIC REGRESSION

What kinds of guarantees do we have about a solution obtain from minimizing the cross-entropy loss function for a logistic regression model?

- Naturally, to answer this, we need to analyze the behavior of the Hessian for this loss function.
- First, we would need to derive the Hessian matrix. **Homework**.
- When we do this, we will find that when the **design matrix *X* has full-column rank** (i.e., its columns are linearly independent), which is the same requirement we have for linear regression, as well as all the **model predictions satisfy $\boldsymbol{h_\theta(x) \in (0, 1)}$** (as opposed to $h_\theta(x) \in [0,1]$; i.e., there are no values of exactly 0 or 1), then the Hessian of the cross-entropy loss will be positive definite! **Proof as Homework**.
    - Notice that X will have full-column rank when it has no features that are co-linear (which should be easily satisfiable and is the same thing we require for linear regression). Similarly, the requirement that $h_\theta(x) \in (0,1)$ will be satisfied if there are instances of datapoints from both target classes (the only way that $h_\theta(x) = 1$ or $h_\theta(x) = 0$ is if the model was trained in the presence of only one of the target classes).
- However, it bears mentioning that if neither of the two assumptions mentioned above are satisfied, then the best one can show is that the Hessian matrix is positive semi-definite.

CLOSED-FORM SOLUTION FOR LOGISTIC REGRESSION?

Now that we have discussed the conditions under which we know that the Hessian matrix will be positive definite (implying that the cross-entropy function is strongly convex), and by extension the solution to minimizing the cross-entropy loss function will be unique, a natural question is:

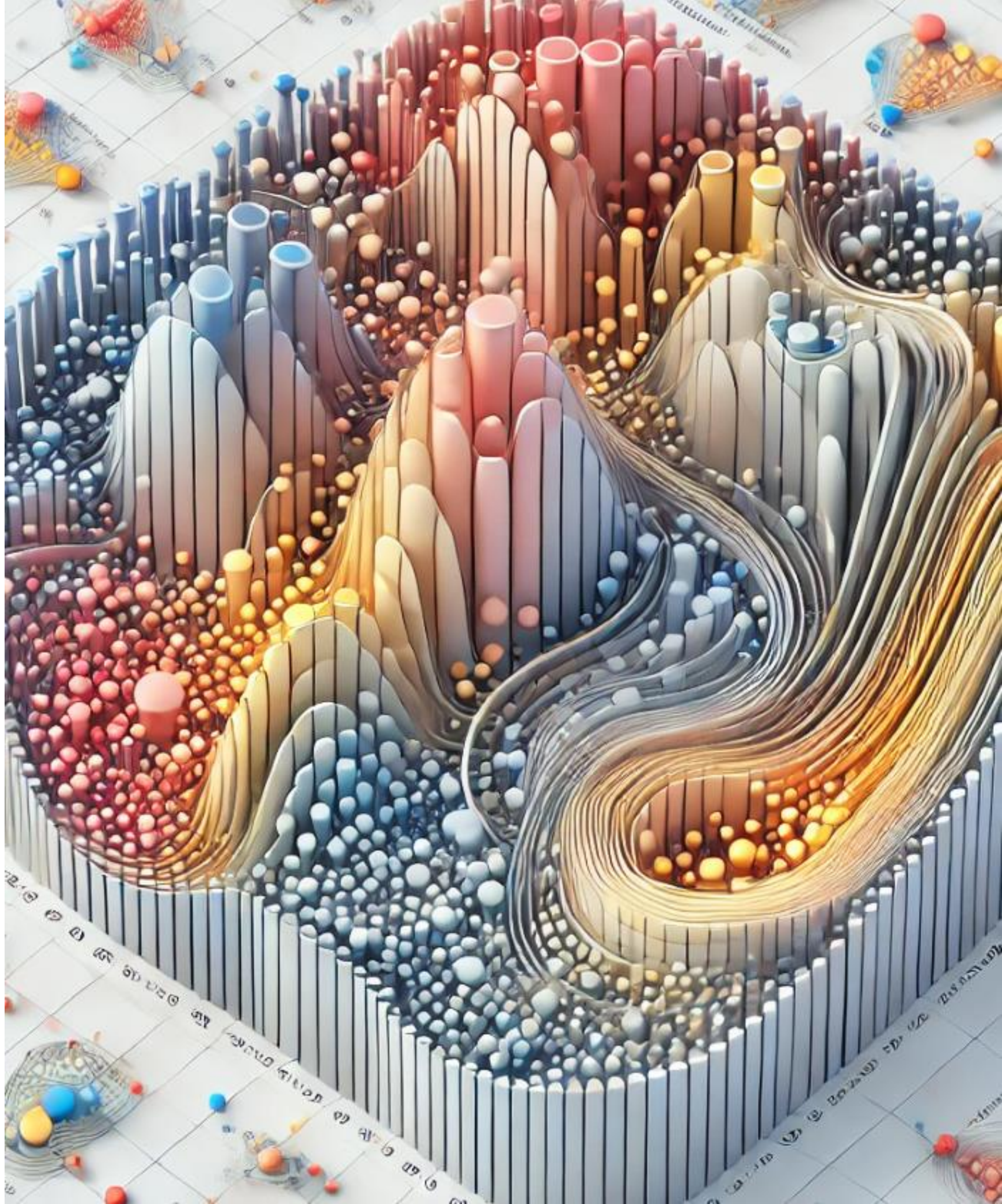
"Can we derive a closed form expression for the solution to logistic regression, similar to what the Normal equations are for linear regression?"

No… Why?

Answer: Because of the **nonlinearity** of the sigmoid function σ.

Thus, **logistic regression** models are essentially always **trained via some optimization algorithm** (typically some form of Newton's method since the cross-entropy loss function will likely be strongly convex).

MULTI-CLASS CLASSIFICATION



GENERAL MULTI-CLASS CLASSIFICATION

Although we have introduced the classical form of logistic regression as a way of classifying binary data, a problem of interest that often arises is when we have more than two target classes. This is referred to as **Multi-Class Classification**.

- In this problem setup, our target variable would have k classes, i.e., $y^{(i)} \in \{1,2,\dots,k\}$, for $k > 2$.
- We would then typically choose a function that would give a numeric output corresponding to the "affinities" for each of the k classes (this would be a vector in $\mathbb{R}^k$).
- Lastly, since we want a model that predicts a single target class, we need a way of converting the vector of affinities to actual probability values (each value corresponding to the probability of observing the corresponding target class). This is analogous to what the sigmoid function does in the binary case.

The Softmax Function

Say that we have some affinity vector $z \in \mathbb{R}^k$ with each element corresponding to one of the target classes. To predict a probability distribution over the k classes, we can utilize the **softmax** function, formally defined as:

$$\text{softmax}([z_1, z_2, \dots, z_k]) \coloneqq \left[\frac{\exp(z_1)}{\sum_j \exp(z_j)}, \frac{\exp(z_2)}{\sum_j \exp(z_2)}, \dots, \frac{\exp(z_k)}{\sum_j \exp(z_k)}\right] \triangleq [\phi_1, \phi_k, \dots, \phi_k].$$

Where $\phi_j \in [0,1]$ are probabilities such that $\sum_j \phi_j = 1$. One could then choose the class j with the largest ϕ_j.

MULTI-CLASS LOGISTIC REGRESSION

With this framework for multi-class classification, we can modify the classical logistic regression model for binary classification with the softmax function instead of the sigmoid function to now generate probabilities for all k target classes.

Multi-Class Logistic Regression

Using the same linear model of $\theta^T x$, the parameters we are interested in would now be a matrix $\theta \in \mathbb{R}^{(n+1)\times k}$, with each row corresponding to the $n+1$ features and each column corresponding to one of the k target classes. This way the function $\theta^T x : \mathbb{R}^{(n+1)} \to \mathbb{R}^k$ returns an affinity vector of dimension k, with each of its elements associated with one of the target classes. Then, using the softmax function, we can convert these affinities to probabilities for each class. That is, we can define our **multi-class logistic regression model** as

$$h_\theta(x) = \text{softmax}(\theta^T x).$$

We can then choose the target class prediction to be the value j such that ϕ_j is the largest probability among the entries in the softmax vector.

TRAINING A MULTI-CLASS LOGISTIC REGRESSION MODEL

Similar to how we trained the logistic regression model for binary classification tasks, we can again train the multi-class model via maximum likelihood. However, our target variable $y^{(i)}$ will now have a **Multinomial distribution** as opposed to a Bernoulli distribution (the multinomial distribution is essentially just a generalization of the Bernoulli distribution to multiple classes). That is, we can express the target variable as

$$\mathbb{P}\left(y^{(i)} \middle| x^{(i)}\right) = \prod_{j=1}^{k} \phi_j^{\mathbb{I}[y^{(i)}=j]},$$

where $\mathbb{I}[C] \in \{0,1\}$ denotes the "**indicator**" function and returns a value of 1 if the condition C is true and a value of 0 if C is false.

The **likelihood function for the multi-class logistic regression** model would be given by

$$L(\theta) = L\left(\theta; \left\{\left(x^{(i)}, y^{(i)}\right)\right\}_{i=1}^{m}\right) = \prod_{i=1}^{m} \mathbb{P}\left(\left(y^{(i)} | x^{(i)}\right); \theta\right) = \prod_{i=1}^{m} \prod_{j=1}^{k} \phi_j^{\mathbb{I}[y^{(i)}=j]} = \prod_{i=1}^{m} \phi_{y^{(i)}}^{(i)}.$$

One can then apply similar methods beyond this point (compute the log-loss, gradient, etc.). **Homework**.

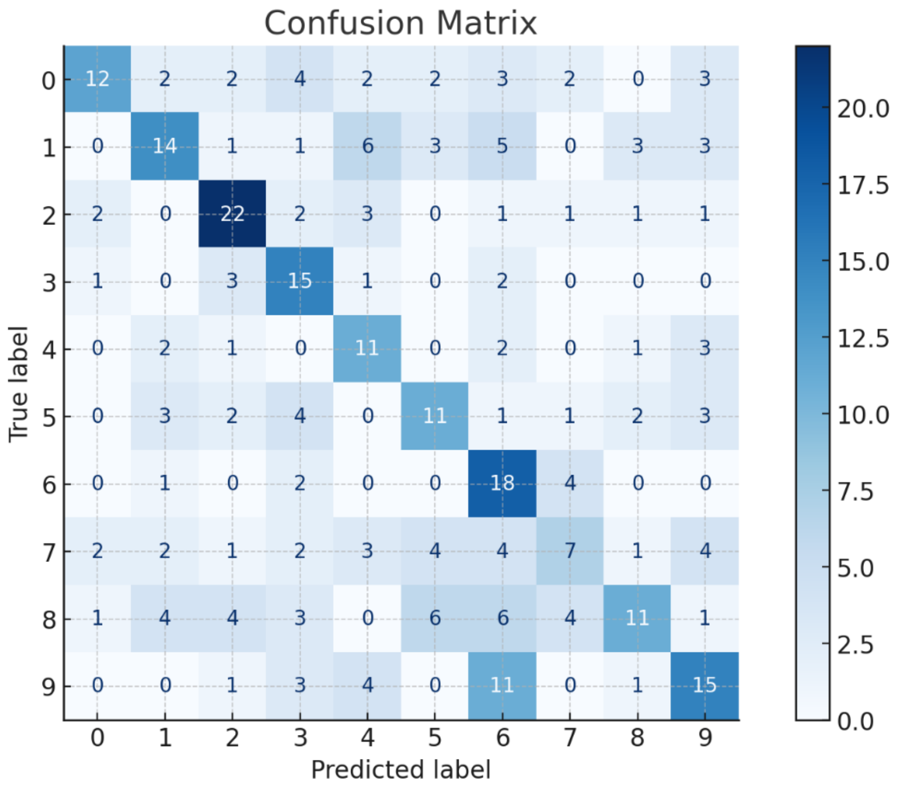
EVALUATION METRICS FOR GENERAL CLASSIFICATION MODELS

Confusion Matrix

In classification problems with K target classes, one way of evaluating a model's performance is to analyze the confusion matrix. This is a $K \times K$ matrix where each row corresponds to the actual target class and each column corresponds to the predicted target class from the model.

As such, the values along the diagonal of the matrix represent how accurate the model is on each of the K classes (the ratio of correct predictions of that class and the total number of actual values of that class).

In this way, one can gain some insight into which target classes the model does well predicting, which it struggles with, and perhaps if the model consistently misclassifies one target class as another.

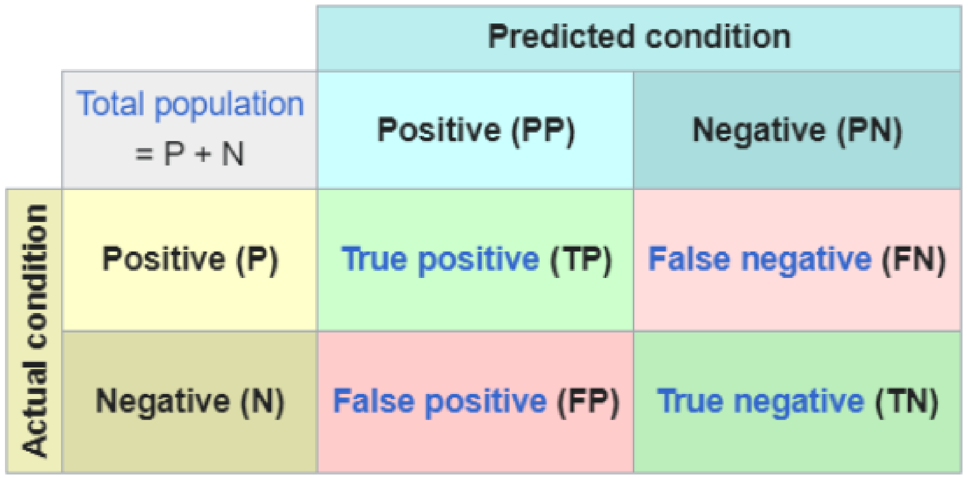


	0	1	2	3	4	5	6	7	8	9
0	12	2	2	4	2	2	3	2	0	3
1	0	14	1	1	6	3	5	0	3	3
2	2	0	22	2	3	0	1	1	1	1
3	1	0	3	15	1	0	2	0	0	0
4	0	2	1	0	11	0	2	0	1	3
5	0	3	2	4	0	11	1	1	2	3
6	0	1	0	2	0	0	18	4	0	0
7	2	2	1	2	3	4	4	7	1	4
8	1	4	4	3	0	6	6	4	11	1
9	0	0	1	3	4	0	11	0	1	15

EVALUATION METRICS FOR BINARY CLASSIFICATION MODELS

- **Accuracy:** The proportion of the number of correct predictions that the model made to the total number of predictions, given by $\frac{TP+TN}{P+N}$.
- **Precision:** Also known as the positive predicted value, this is the proportion of the true positive predictions of the model to the total positive predictions that the model made, given by $\frac{TP}{PP}$.
    - A high precision indicates that when a model predicted a positive value, it is likely to be correct. This metric is crucial when the cost of false positives is high, or one wants to be certain that the positive predictions are accurate.
- **Recall (Sensitivity):** Also known as the true positive rate (TPR), this is the proportion of true positive predictions of the model to the total actual positive predictions, given by $\frac{TP}{P}$. A high recall indicates that the model is good at capturing all the positive instances.
    - This is a crucial metric when missing a positive instance has significant consequences and you want to ensure that as many positives as possible are captured, even if it means a few false positives might be included.
- **F1-Score:** This is a metric defined as $2 \cdot \frac{Precision \times Recall}{Precision + Recall}$ that combines both precision (a measure of the false positives) and recall (a measure of the false negatives) into a unified score via the harmonic mean.

2×2 Confusion Matrix



		Predicted condition	
	Total population = P + N	Positive (PP)	Negative (PN)
Actual condition	Positive (P)	True positive (TP)	False negative (FN)
	Negative (N)	False positive (FP)	True negative (TN)

RECEIVER OPERATING CHARACTERISTIC (ROC) CURVE FOR BINARY CLASSIFICATION MODELS

The Receiver Operating Characteristic (ROC) Curve

The ROC curve is a graphical representation used to assess the performance of a **binary classification** model. It illustrates the trade-off between the **True Positive Rate** (TPR) and the **False Positive Rate** (FPR) over different threshold values. TPR and FPR are defined as:

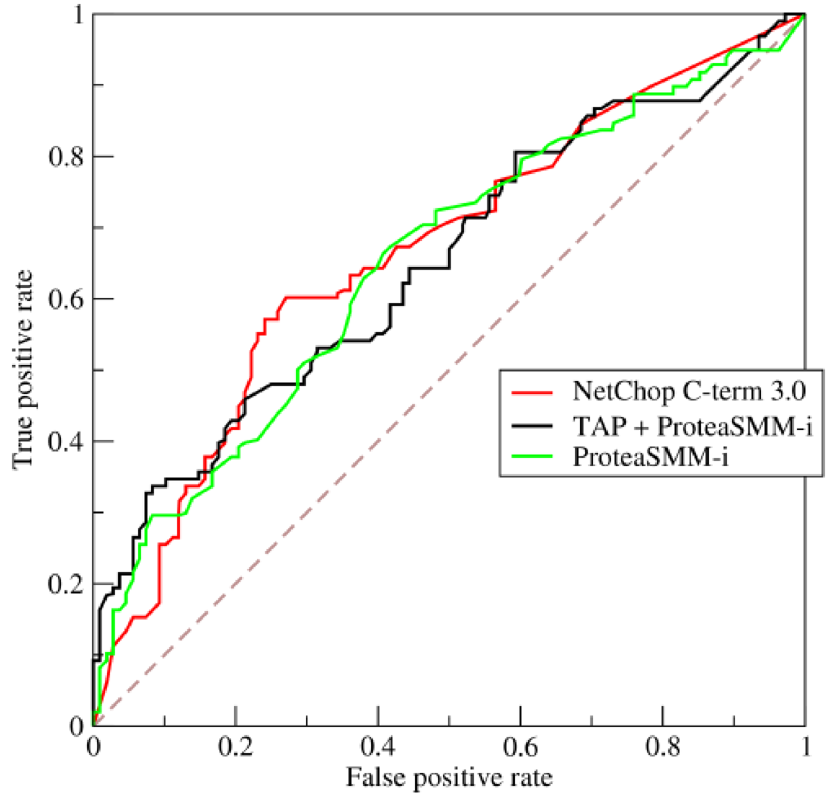
$$TPR = \frac{TP}{TP + FN} = \text{Recall},$$

which measures the proportion of positive values correctly identified by the model.

$$FPR = \frac{FP}{FP + TN},$$

which measures the proportion of negative values that were incorrectly identified by the model. A **Threshold** is the "cut-off" value used to classify a positive value. For example, the model may report a positive value with a 0.7 probability, but there may be a threshold value of acceptance of 0.8, in which case it would not be predicted as a positive value.

- The Area Under the Curve (AUC) is a single-number that summarizes the model's overall performance: AUC=1.0 implies the classifier is always correct, AUC=0.5 implies the classifier has no predictive power (equivalent to randomly guessing).


True positive rate
False positive rate
NetChop C-term 3.0
TAP + ProteaSMM-i
ProteaSMM-i