CS2109S: Introduction to AI and Machine Learning

Lecture 6:
Logistic Regression

18 February 2025

DO NOT CLOSE YOUR POLLEVERYWHERE APP

There will be activities ahead

Announcements

Midterm – Reminder

- Date & Time:
  - **Tuesday, 4 March 2025**, from **18:30** to **20:00**
- Venue:
  - **MPSH 2A & 2B**
- Format:
  - Digital Assessment (**Examplify**)
- Materials:
  - All topics covered **until and including Lecture 6**
- Cheatsheet:
  - **1 x A4 paper, both sides**
- Calculators:
  - **Standard** and **scientific calculators** are allowed.
  - No graphing/programmable calculators.

More details will be announced later.

Midterm – Examplify

All the info:

https://nus.atlassian.net/wiki/spaces/DAstudent/overview

Video

https://mediaweb.ap.panopto.com/Panopto/Pages/Viewer.aspx?id=48df9509-7daf-41f4-9ee8-ae22008a7383

Common briefing:

https://nus.atlassian.net/wiki/spaces/DAstudent/pages/22511675/Common+Briefing+Sessions

Midterm – Coverage

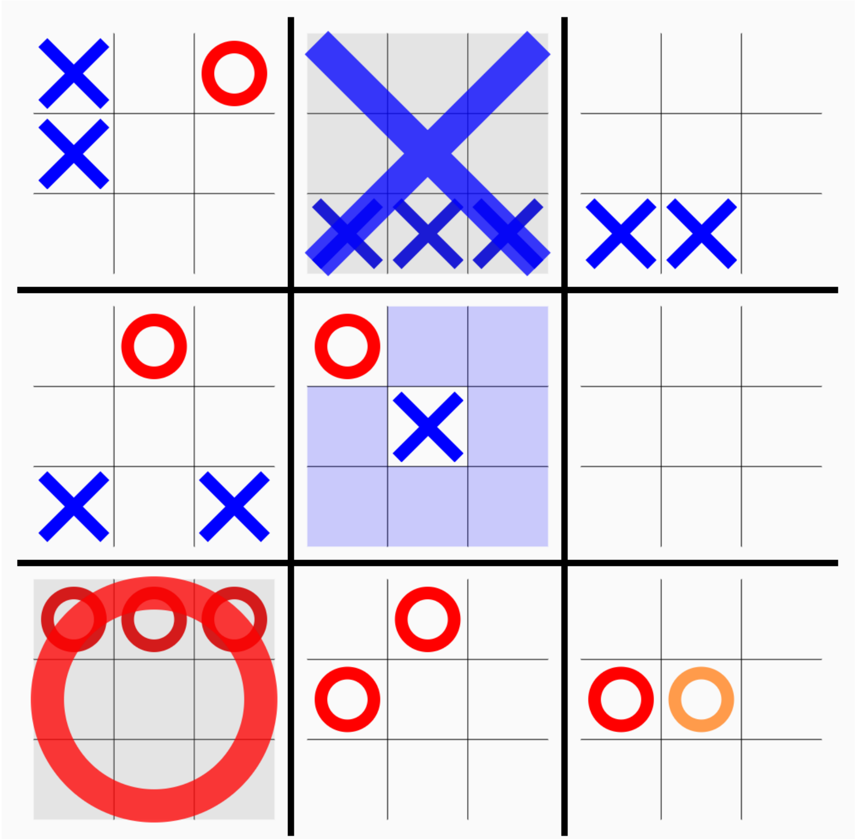
- **Intelligent agents**
- **Uninformed search**: problem formulation, BFS, DFS, UCS, DLS, IDS, visited memory.
- **Informed search**: A* search, heuristics, admissibility, consistency, dominance.
- **Local search**: problem formulation, hill-climbing.
- **Adversarial search**: game tree, minimax, alpha-beta pruning, cutoff.
- **ML, Supervised learning**: hypothesis class, performance measure, learning algorithms
- **Decision trees**: decision tree learning, pruning.
- **Linear regression**: linear model, feature transformations, loss, gradient descent, normal equation.
- **Logistic regression**: “squashed” linear model, multi-class classification, generalization.

Basically, materials covered up to week 6

Schedule: Recess Week & Midterm

- Next week: Recess Week, no tutorial
- Week 7: has Midterm, no tutorial
- Week 8: Lecture 7, has tutorial

Mini-Project



- Develop an agent to play **Ultimate Tic-Tac-Toe.**
- Can use search, machine learning, or both!
- Compete against our agents.
- Details regarding timeline, grading, rules will be given on release.
- Optional 1v1 contest with S$100 total prize!
  - Will not impact grades, just for fun.

Will be released after the midterm!

Lecture

Recap

- Linear Regression: **fitting a line** to data
- Linear Model
  - d dimensional input features: $h_{w}(x) = \sum_{j=0}^{d} w_j x_j = w^T x$
- Finding the best function, i.e., one that minimizes the loss
  - Normal Equation: **set derivative to 0, solve**
  - Gradient Descent
    - Gradient Descent Algorithm: **follow –gradient** to reduce error
    - Linear Regression with Gradient Descent: **convex** optimization, **one minimum**
    - Problem: Features of Different Scales: **normalize!**
    - Variants of Gradient Descent: batch, mini-batch, stochastic

Outline

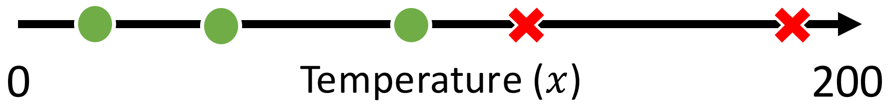
- Logistic Regression
  - Data
  - Model
  - Loss
- Learning via Gradient Descent
- Multi-Class classification
- Advanced Topics in Supervised Learning
  - Generalization, Dataset, Model Complexity
  - Overfitting & Underfitting
  - Hyperparameter Tuning

Outline

- **Logistic Regression**
  - Data
  - Model
  - Loss
- Learning via Gradient Descent
- Multi-Class classification
- Advanced Topics in Supervised Learning
  - Generalization, Dataset, Model Complexity
  - Overfitting & Underfitting
  - Hyperparameter Tuning

Example: Engine Failure Prediction

Temperature	Failure?
20	False
50	False
95	False
120	True
190	True
…	…



Will engine fail at 150°C?

Data

Suppose:

- We are given N data points.
- Each data point consists of features and a target variable.
- The features are described by a vector of real numbers in dimension d.
- The target is {0,1}, where 0 is "negative" class and 1 is "positive" class.

Data – Math

Suppose:

$$D = \{(x^{(1)}, y^{(1)}), (x^{(2)},\ y^{(2)}), \dots, (x^{(N)},\ y^{(N)})\},$$

where for all $i \in \{1, \dots, N\}$

$$\text{Features: } x^{(i)} \in \mathbb{R}^{d}$$

$$\text{Targets: } y^{(i)} \in \{0,1\}$$

Task

Suppose we are given another data point $x \in \mathbb{R}^d$ and no target. Based on the dataset, find a function that predicts the target $y \in \{0,1\}$ for that x.

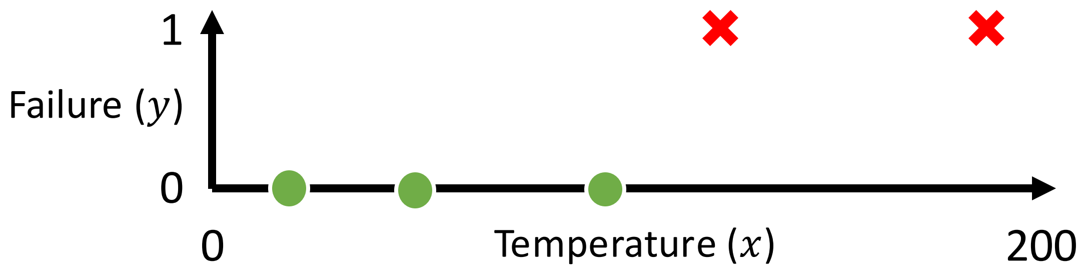
This task is called classification.

Since the target has only two classes, this is called **binary classification**.

Example: Engine Failure Prediction

Dataset D

i	$x^{(i)}$	$y^{(i)}$
1	20	0
2	50	0
3	95	0
4	120	1
5	190	1
…	…	…



x = 150°C

Will engine fail at x?

What class of functions should we use?

"Squashing" of the Linear Model

Let's define a hypothesis class that is the set of "squashed" linear functions.

What are linear functions that map as follows?

- **From** d-dimensional vectors of real numbers
- **To** 1-dimensional real numbers (scalars)

We know this already! $w_0 x_0 + w_1 x_1 + w_2 x_2 + \cdots + w_d x_d$

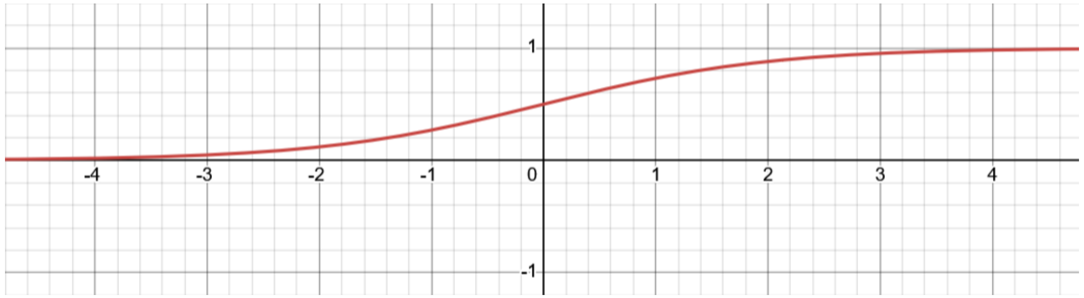
What function can "squash" our linear functions, so it outputs within $[0,1]$?

Bonus point if it is differentiable!

Background: Sigmoid Function

The sigmoid function is a mathematical function that maps a real number to a value between 0 and 1.

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$



This function is differentiable, with derivative:

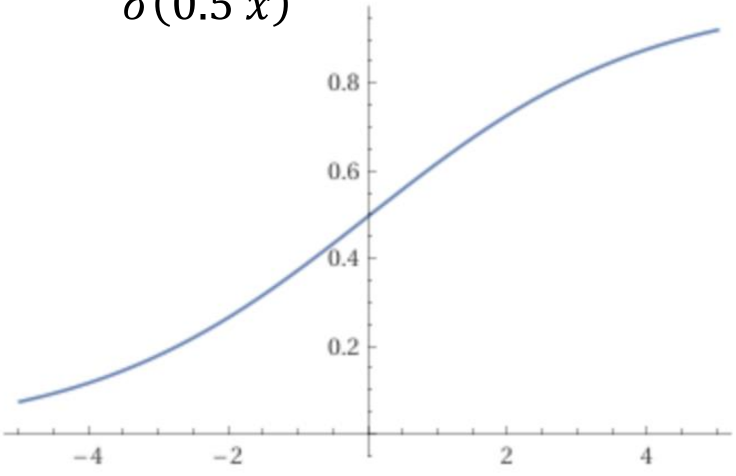
$$\sigma'(x) = \sigma(x)(1 - \sigma(x))$$

Also known as the **logistic function**.

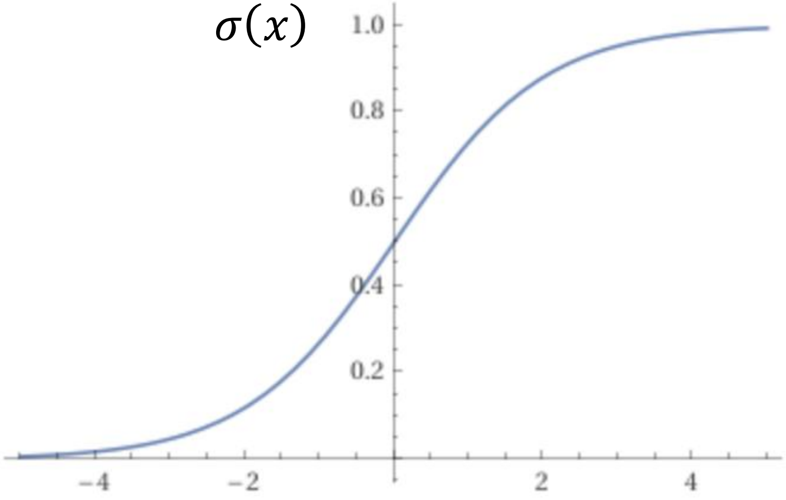
Background: Sigmoid Function

Plot $\sigma(x) = \frac{1}{1+e^{-x}}$

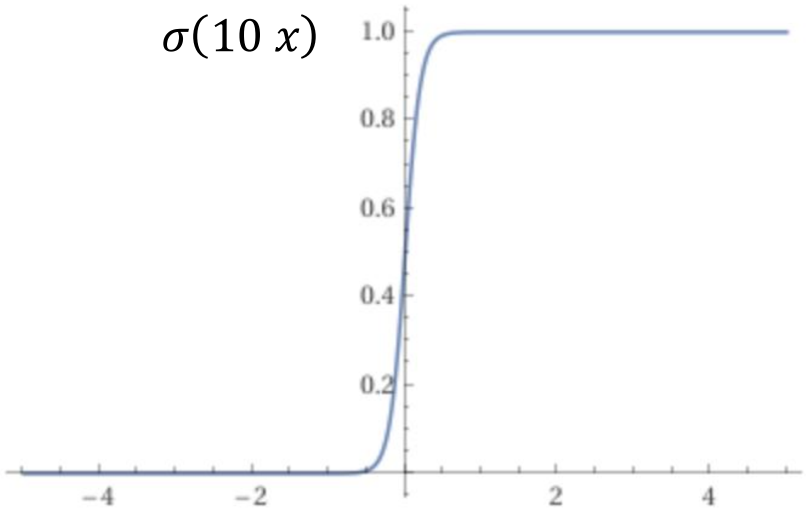
$\sigma(0.5\, x)$



$\sigma(x)$



$\sigma(10\, x)$



How does the curve of $\sigma(c+x)$ change compared to $\sigma(x)$, where $c>0$ is a constant?

Poll Everywhere

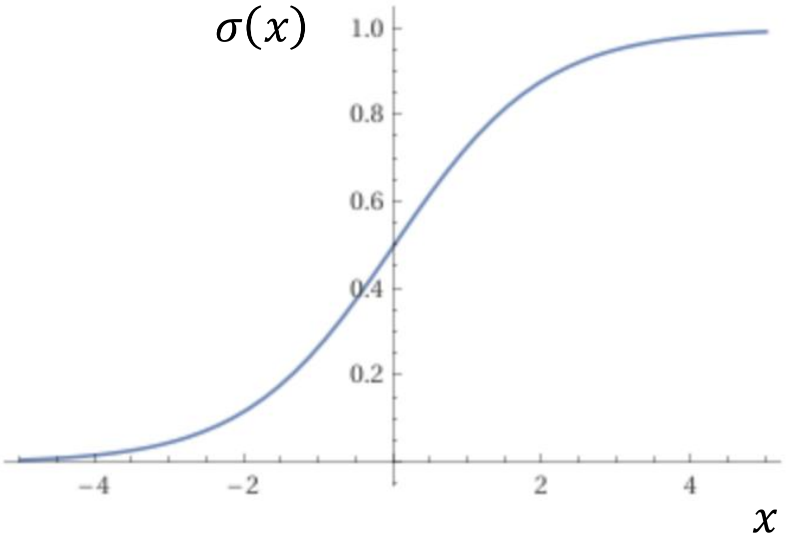
How does the curve of $\sigma(c + x)$ change compared to $\sigma(x)$, where $c > 0$ is a constant?

A. Shifts up

B. Shifts down

C. Shifts left

D. Shifts right



Poll Everywhere

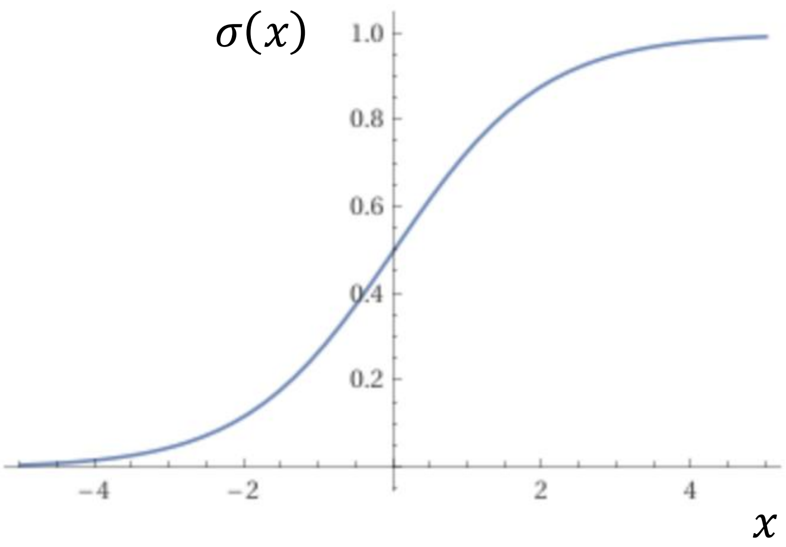
How does the curve of $\sigma(c + x)$ change compared to $\sigma(x)$, where $c > 0$ is a constant?

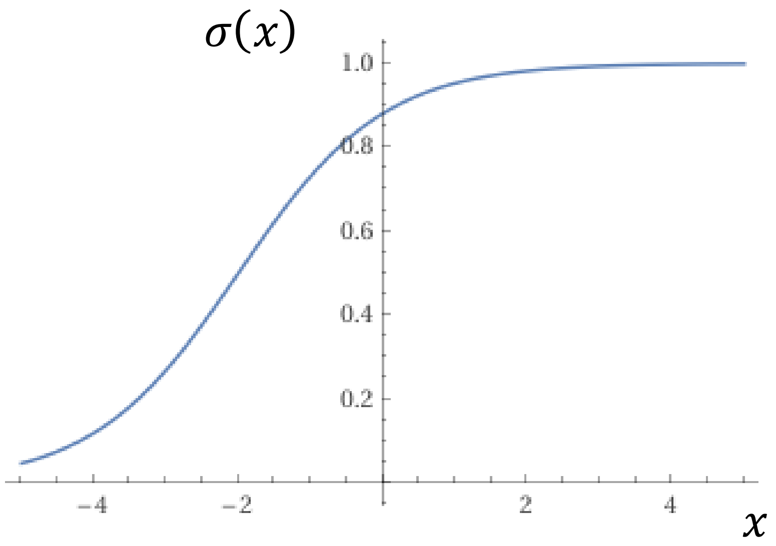
A. Shifts up

B. Shifts down

C. Shifts left

D. Shifts right





Logistic Regression Model

Given an input vector x of dimension d, the hypothesis class of linear models is defined as the set of functions:

$$h_w(x) = \sigma(w_0 x_0 + w_1 x_1 + w_2 x_2 + \cdots + w_d x_d)$$

where $w_0, \dots, w_d$ are **parameters/weights** and $x_0 = 1$ is a dummy variable.

We shorthand this function by using the dot product:

$$h_w(x) = \sigma(w^T x)$$

Background: Probability

- **Probability** $P(X)$: A measure of the likelihood that an event X will occur. It ranges from 0 (impossible) to 1 (certain).
  - Example: $P(\text{Failure} = true) = 0.01$ means that the probability of engine failure (i.e., that engine failure is true) is 1%.

- **Conditional probability** $P(Y|X)$ is the probability of an event Y occurring given that an event X has already occurred.
  - Example: $P(\text{Failure} = true|\text{Temperature} = 90) = 0.8$ means that the probability of engine failure (i.e., that engine failure is true) given that the temperature is 90° C is 80%.

Classification with Logistic Regression Model

- Logistic regression model outputs a real number in range [0,1]
- The output of this model can be treated as the probability of an input to be of class 1 (confidence score).
  - Example: engine failure prediction model
    $$h_w(x) = P(\text{Failure} = true|x) = 0.8$$
    The model predicts that the probability of engine failure is 80%
- To decide whether an input belongs to a certain class, compare the probability output to a decision threshold.
  - Example: decision threshold = 0.5 → if the probability is greater than 0.5, classify the input as class 1; otherwise, classify it as class 0.

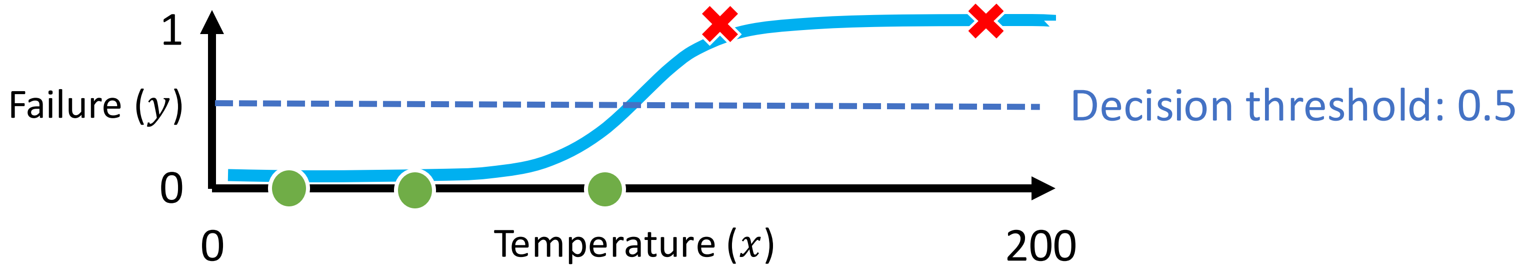
Example: Engine Failure Prediction

$$h_w(x) = \sigma(w_0 x_0 + w_1 x_1)$$

$x_0 = 1$ (dummy variable)

Dataset D

i	$x^{(i)}$	$y^{(i)}$
1	20	0
2	50	0
3	95	0
4	120	1
5	190	1
…	…	…



$$h_w(x) = \sigma(-100 + 1x_1)$$

We can set the threshold to 0.5:

- When $h_w(x) \geq 0.5$, output the classification of x as 1 (fail)
- When $h_w < 0.5$, output the classification of x as 0 (not fail)

Decision Boundary

- A **decision boundary** is a surface (or line in two dimensions, or hyperplane in >2 dimensions) that separates the different classes in the feature space.
- It is an <u>intersection</u> between the hypothesis function and the decision threshold.
- It defines the points where the classification model changes its predicted class from one to another.
  - For example, in a binary classification task, the decision boundary divides the feature space into two regions, each corresponding to one of the two classes.

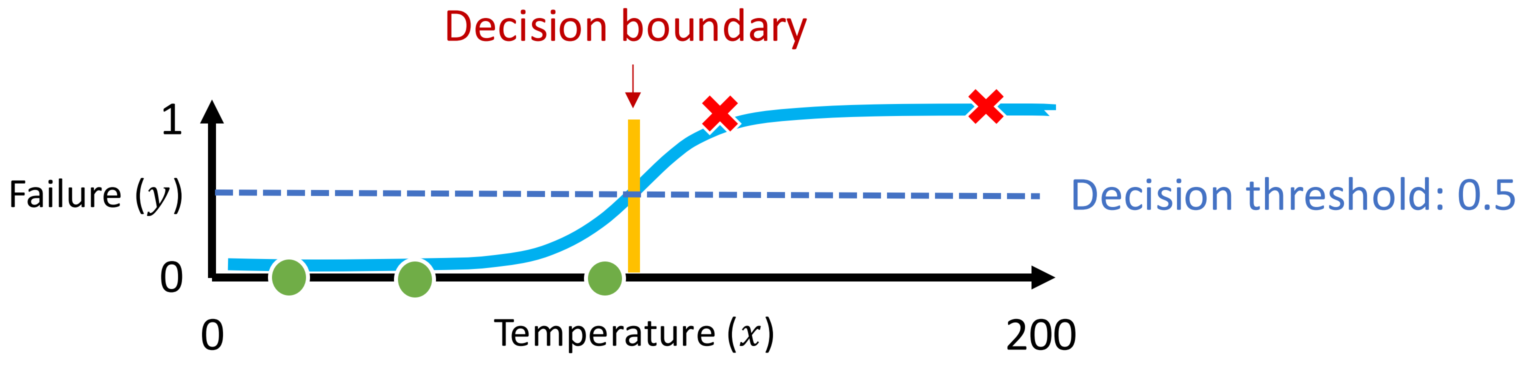
Example: Engine Failure Prediction

$$h_w(x) = \sigma(w_0 x_0 + w_1 x_1)$$

$x_0 = 1$ (dummy variable)

Dataset D

i	$x^{(i)}$	$y^{(i)}$
1	20	0
2	50	0
3	95	0
4	120	1
5	190	1
…	…	…



$$h_w(x) = \sigma(-100 + 1x_1)$$

Example: Engine Failure Prediction

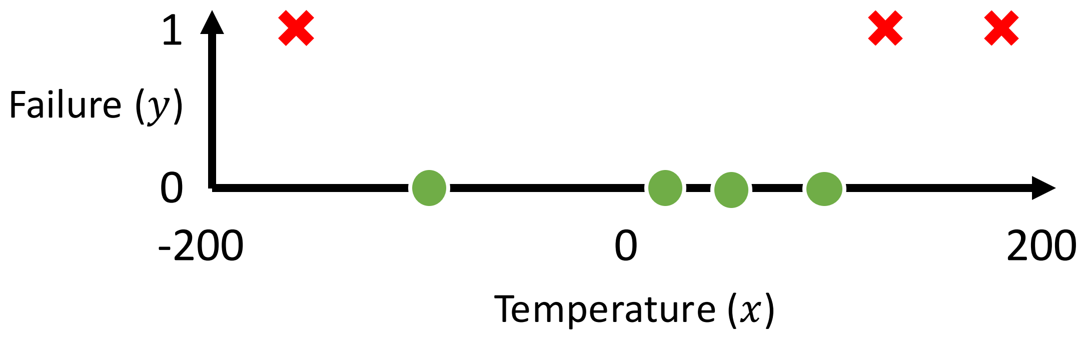
with low-temperature failures

$$h_w(x) = \sigma(w_0 x_0 + w_1 x_1)$$

$x_0 = 1$ (dummy variable)

Dataset D

i	$x^{(i)}$	$y^{(i)}$
1	20	0
2	50	0
3	95	0
4	120	1
5	190	1
6	-100	0
7	-170	1
…	…	…



Non-Linearly Separable Data

- Data is said to be **non-linearly separable** when a single linear decision boundary (like a straight line in 2D or a hyperplane in higher dimensions) cannot effectively separate the different classes.
- The decision boundary required to separate the classes needs to be non-linear or more complex.
  - Use non-linear (more complex) models.
  - Use **non-linear feature transformations.**

Example: Engine Failure Prediction

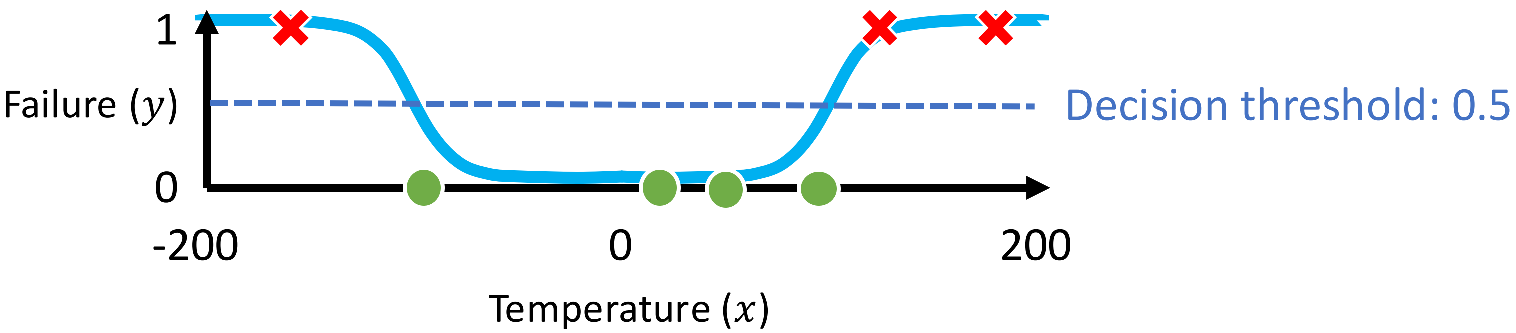
with low-temperature failures

$$h_w(x) = \sigma(w_0 x_0 + w_1 x_1)$$

$x_0 = 1$ (dummy variable)

Dataset D

i	$x^{(i)}$	$y^{(i)}$
1	20	0
2	50	0
3	95	0
4	120	1
5	190	1
6	-100	0
7	-170	1
…	…	…



x_1	-200	-100	0	100	200	
x_1'	-1	-0.5	0	0.5	1	Scale to between -1 and 1
x_2	1	0.25	0	0.25	1	Transform: $x_2 = (x_1')^2$

$$h_w(x) = \sigma(-5 + 20x_2)$$

Recall – Linear Regression: Measuring Fit

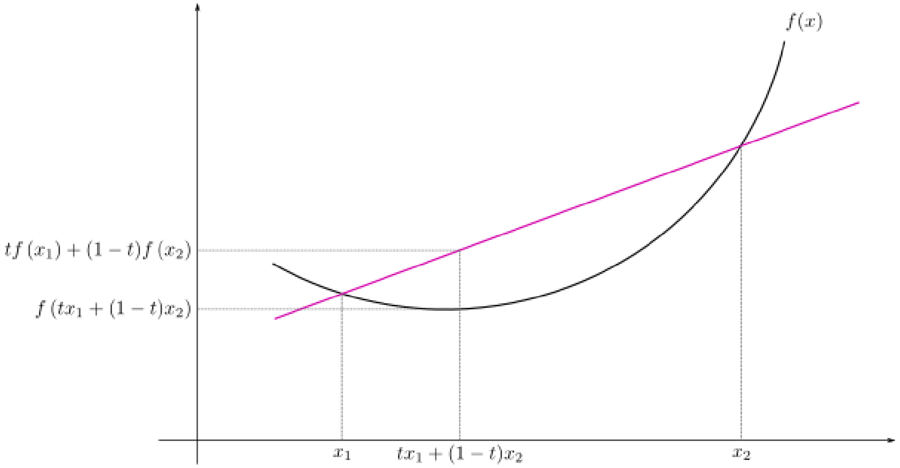
For N examples $\{(x^{(1)}, y^{(1)}), \ldots, (x^{(N)}, y^{(N)})\}$, define **mean squared error (MSE)**:

$$J_{MSE}(w) = \frac{1}{2N}\sum_{i=1}^{N}\left(h_w\left(x^{(i)}\right) - y^{(i)}\right)^2$$

- We want to find w that minimize this loss function!
- Can we use MSE for logistic regression? Any issue?

Recall – Background: Convexity

- A real-valued one-dimensional function is called convex if the line segment between any two distinct points on the graph of the function lies above or on the graph between the two points.



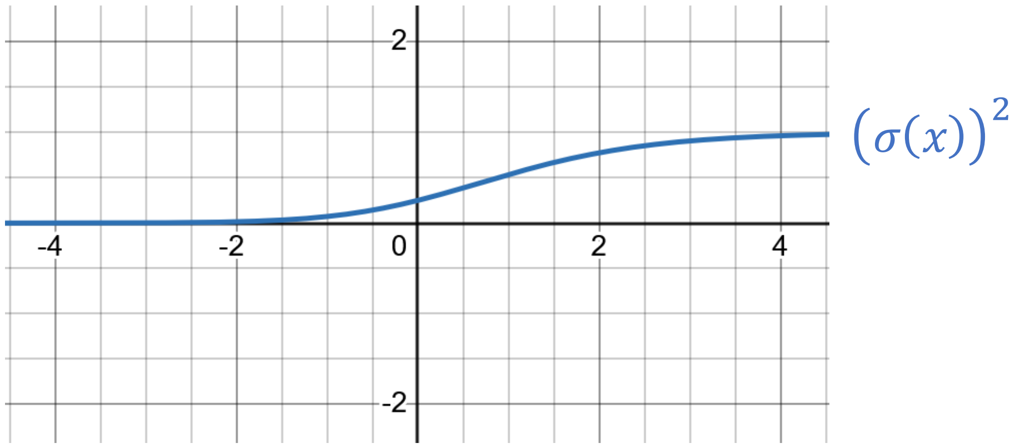
- For function with two or more inputs, think of bowl-shaped landscape.

Logistic Regression with MSE

Theorem: MSE loss function is non-convex for logistic regression.

Proof Idea:

- Logistic function $\sigma(x) = \frac{1}{1+e^{-x}}$ is non-convex.
- Squaring of $\sigma(x)$, i.e., main component of MSE, is non-convex–it retains the "S" shape.

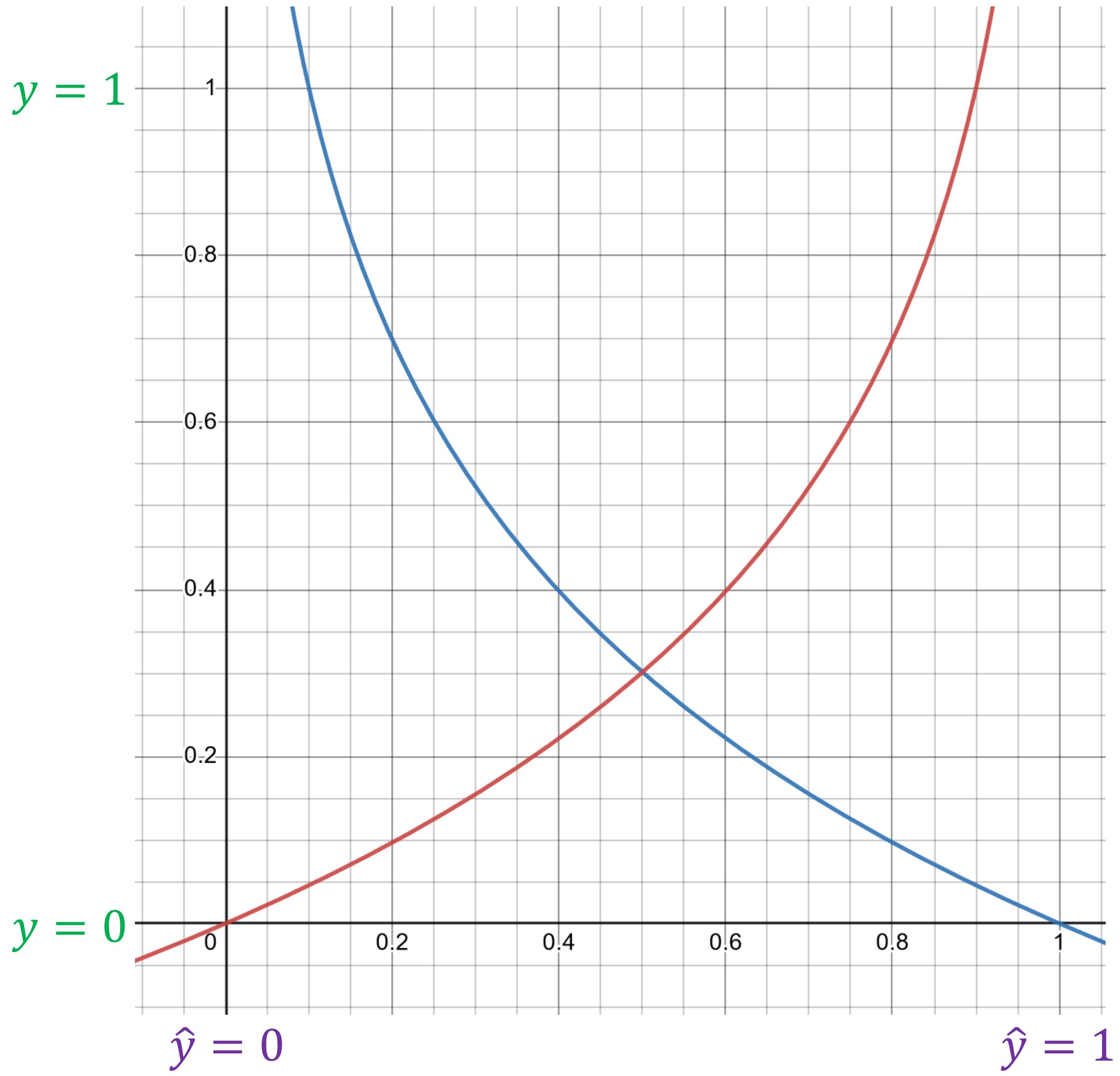


Binary Cross Entropy (BCE)

Given a probability value $y \in [0,1]$ and $\hat{y} \in [0,1]$, the difference between these probability values can be computed as follows.

$$BCE(y, \hat{y}) = -y\log(\hat{y}) - (1 - y)\log(1 - \hat{y})$$

Binary Cross Entropy (BCE)



If true value y and prediction $\hat{y}$ are **close**, BCE loss is small.

- 0 if $y = \hat{y}$.

If true value y and prediction $\hat{y}$ are **far**, BCE loss is large.

- $\to \infty$ if $y = 0, \hat{y} \to 1$, and vice versa.

Hence, it is a **good** loss function

" $\to$ " means approaching

Binary Cross Entropy (BCE) Loss

For N examples $\{(x^{(1)}, y^{(1)}), \ldots, (x^{(N)}, y^{(N)})\}$, we can define the **mean BCE**:

$$J_{BCE}(w) = \frac{1}{N}\sum_{i=1}^{N} BCE\left(y^{(i)}, h_w(x^{(i)})\right)$$

This is known as **binary cross entropy (BCE) loss** or **logistic loss** (log loss).

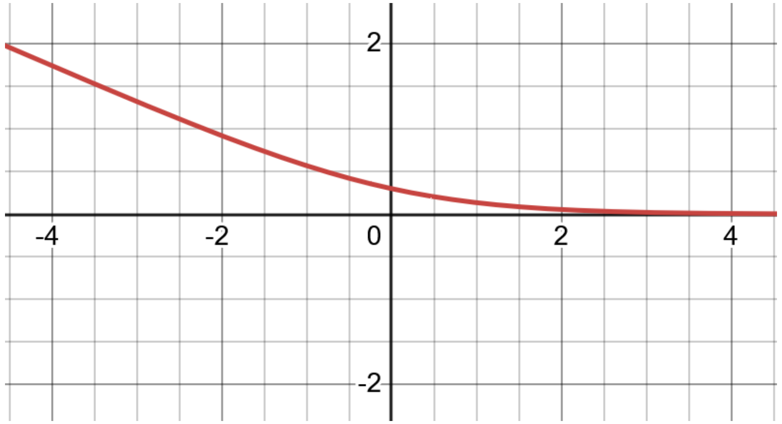
Logistic Regression with BCE Loss

Theorem: BCE loss function is convex for logistic regression.

- There is only one minimum which is the global minimum.

Proof Idea:

- Logistic function $\sigma(x) = \frac{1}{1+e^{-x}}$ is non-convex.
- However, $-\log$ (i.e., main component of BCE) of $\sigma(x)$ is convex.



$-\log(\sigma(x))$

Outline

- Logistic Regression
  - Data
  - Model
  - Loss
- **Learning via Gradient Descent**
- Multi-Class classification
- Advanced Topics in Supervised Learning
  - Generalization, Dataset, Model Complexity
  - Overfitting & Underfitting
  - Hyperparameter Tuning

Recall – Gradient Descent

- Start at some w (e.g., randomly initialized).
- Update w a step to the opposite direction of the gradient (i.e., towards lower loss)

$$w_j \leftarrow w_j - \underbrace{\gamma}_{\text{Learning Rate}} \frac{\partial J(w_0, w_1, \ldots)}{\partial w_j}.$$

- Learning rate $\gamma > 0$ is a hyperparameter that determines the step size
- Repeat until termination criterion is satisfied.
  - E.g., change between steps is small, maximum number of steps is reached, etc

Logistic Regression with Gradient Descent

Hypothesis:

$$h_w(x) = \sigma(w_0 + w_1 x_1 + w_2 x_2)$$

Loss Function:

$$J_{BCE}(w) = \frac{1}{N}\sum_{i=1}^{N} BCE(y^{(i)}, h_w(x^{(i)}))$$

Weight Update:

$$w_j \leftarrow w_j - \gamma \frac{\partial J_{BCE}(w_0, w_1, \ldots)}{\partial w_j}$$

Derivative:

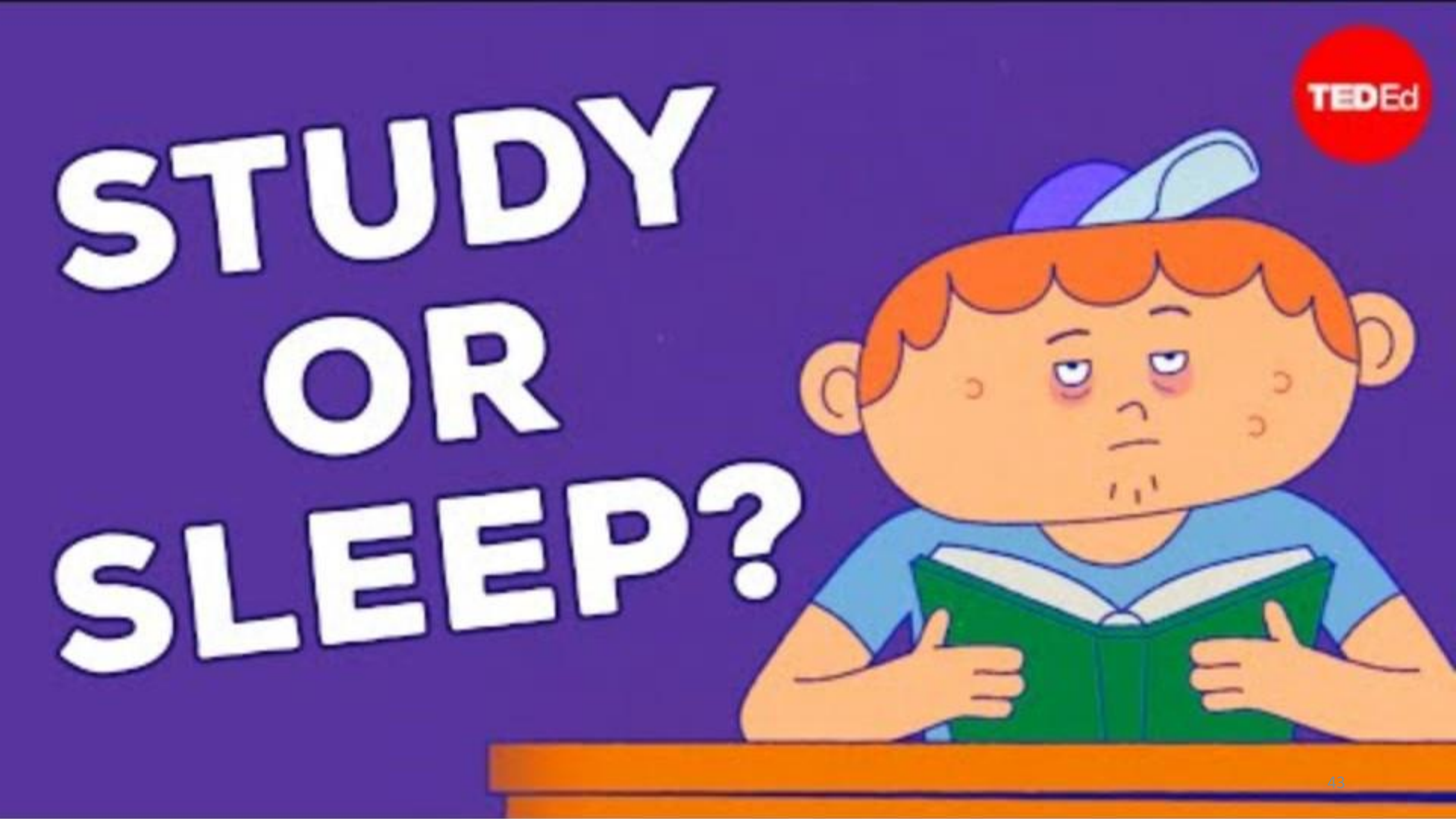
$$\frac{\partial J_{BCE}(w)}{\partial w_j} = \frac{\partial}{\partial w_j}\frac{1}{N}\sum_{i=1}^{N} BCE(y^{(i)}, h_w(x^{(i)}))$$

$$= \frac{1}{N}\sum_{i=1}^{N}(h_w(x^{(i)}) - y^{(i)})\, x_j^{(i)}$$

Same as Linear Regression!

Will discuss the derivations in tutorial

Break


STUDY
OR
SLEEP?
TED Ed

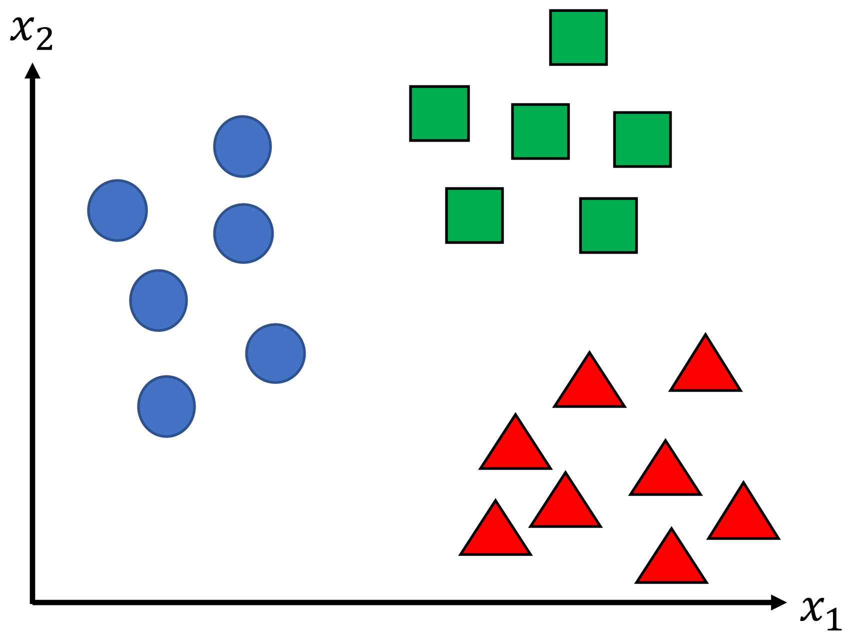
Outline

- Logistic Regression
  - Data
  - Model
  - Loss
- Learning via Gradient Descent
- **Multi-Class classification**
- Advanced Topics in Supervised Learning
  - Generalization, Dataset, Model Complexity
  - Overfitting & Underfitting
  - Hyperparameter Tuning

Example: Animal Prediction

Given a set of features describing an animal (e.g., x_1 weight, x_2 height),

Predict whether it is a cat, dog, or rabbit.



Multi-class Classification

Suppose:

- We are given N data points.
- Each data point consists of features and a target variable.
- The features are described by a vector of real numbers in dimension d.
- The target is {1, 2, …, C} where C is the number of classes.

Suppose we are given another data point $x \in \mathbb{R}^d$ and no target. Based on the dataset, find a model that predicts the target $y \in \{1,2,\dots,C\}$ for that x.

How to do this if we only have binary classifiers?

Binary to Multi-class Classification

Converting binary classification into **multi-class classification** involves breaking down multi-class classification problem into a set of binary classification problem.

Techniques

- One-vs-one
- One-vs-rest

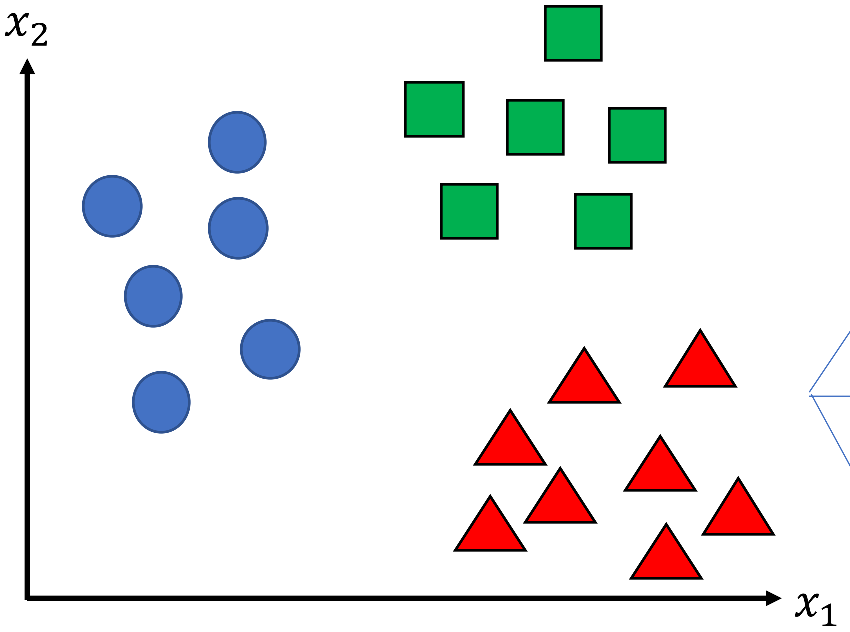
One-vs-One

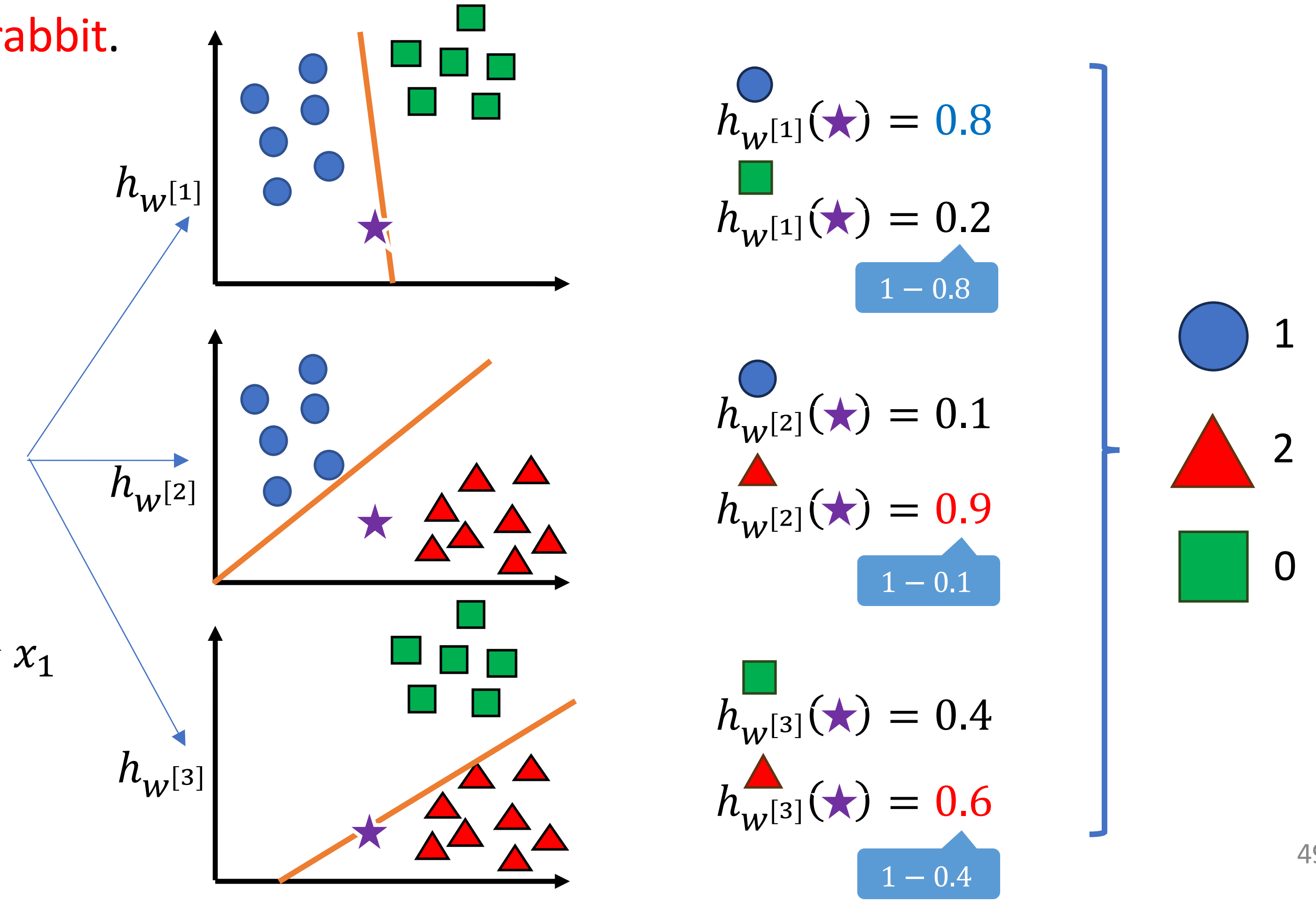
- One-vs-One is a technique where a separate binary classifier is trained for **every pair of classes**. For C classes, this results in $\frac{C(C-1)}{2}$ classifiers. Each classifier distinguishes between two specific classes.
- **Example:** For a 3-class problem with classes 1, 2, and 3:
  - Classifier 1: Distinguish between class 1 and 2.
  - Classifier 2: Distinguish between class 1 and 3.
  - Classifier 3: Distinguish between class 2 and 3.
- During prediction, each classifier votes for a class, and the class with the most votes is selected.

One-vs-One – Example

Given a set of features describing an animal (e.g., x_1 weight, x_2 height),

Predict whether it is a cat, dog, or rabbit.





- Fit one classifier per class pair
- Pick **most wins**

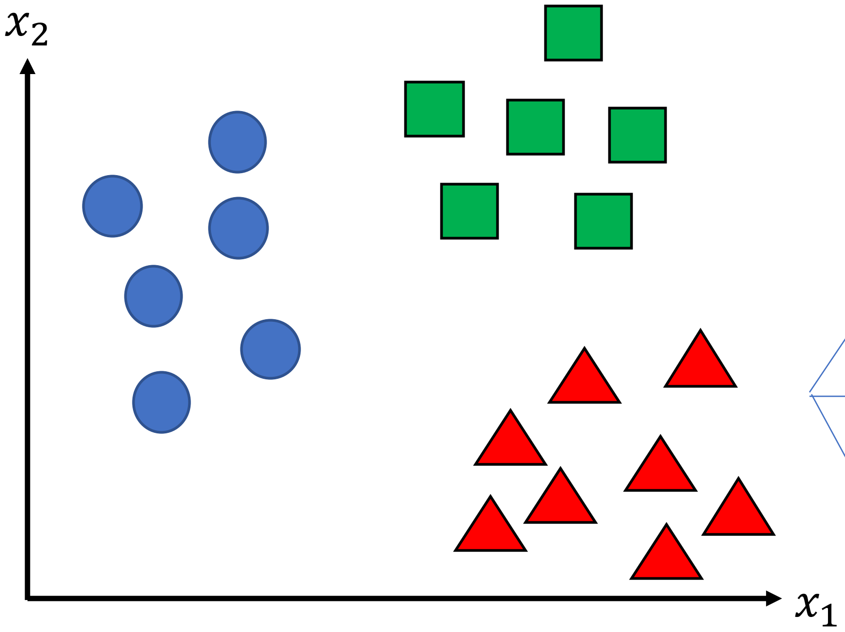
One-vs-Rest

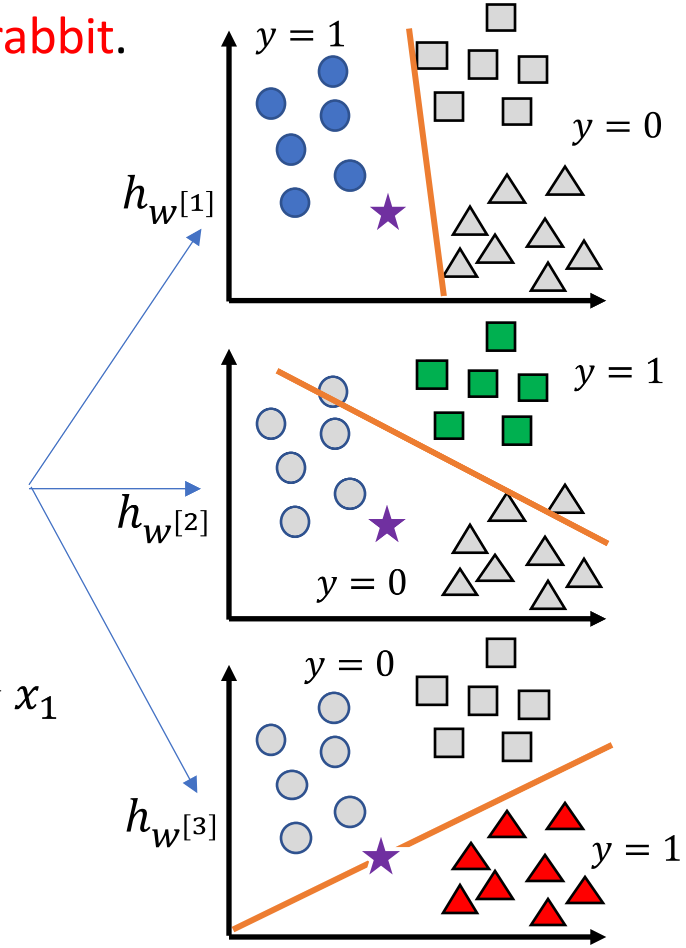
- One-vs-Rest is a technique where a separate binary classifier is trained for each class, **treating all other classes as a single combined class**. For C classes, this results in C classifiers.
- **Example:** For a 3-class problem with classes 1, 2, and 3:
  - Classifier 1: Distinguish between class 1 and not-1 (2 & 3).
  - Classifier 2: Distinguish between class 2 and not-2 (1 & 3).
  - Classifier 3: Distinguish between class 3 and not-3 (1 & 2).
- During prediction, the classifier with the highest probability output (confidence score) determines the class.

One-vs-Rest – Example

Given a set of features describing an animal (e.g., x_1 weight, x_2 height),

Predict whether it is a cat, dog, or rabbit.





$h_{w^{[1]}}(\star) = 0.8$ ✔

$h_{w^{[2]}}(\star) = 0.1$

$h_{w^{[3]}}(\star) = 0.5$

- Fit one classifier per class,
- Fit against all other classes
- Pick **highest probability**

Outline

- Logistic Regression
  - Data
  - Model
  - Loss
- Learning via Gradient Descent
- Multi-Class classification
- **Advanced Topics in Supervised Learning**
  - Generalization, Dataset, Model Complexity
  - Overfitting & Underfitting
  - Hyperparameter Tuning

Generalization

- In supervised learning, and machine learning in general, the model's performance on unseen data is all we care about. This ability to perform well on new, unseen data is known as the model's generalization capability.
- Measuring a model's error is a common practice to quantify the performance of the model. This error, when evaluated on unseen data, is known as the generalization error.
- There are two factors that affect generalization:
  - Dataset quality and quantity
  - Model complexity

Dataset Quality

- **Relevance**: Dataset should contain relevant data, i.e., features that are relevant for solving the problem.
- **Noise**: Dataset may contain noise (irrelevant or incorrect data), which can hinder the model's learning process and reduce generalization.
- **Balance** (for classification): Balanced datasets ensure that all classes are adequately represented, helping the model learn to generalize well across different classes.

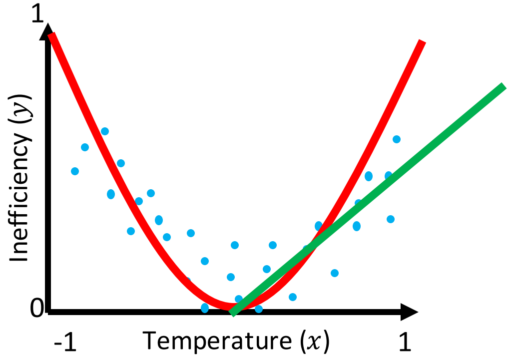
- Basically, Garbage in → Garbage out

Dataset Quantity

- In general, having more data typically leads to better model performance, provided that the model is expressive enough to accurately capture the underlying patterns in the data.
- **Extreme case**: if the dataset contains every possible data point, the model would no longer need to "guess" or make predictions. Instead, it would only need to simply memorize all the data!

Model Complexity

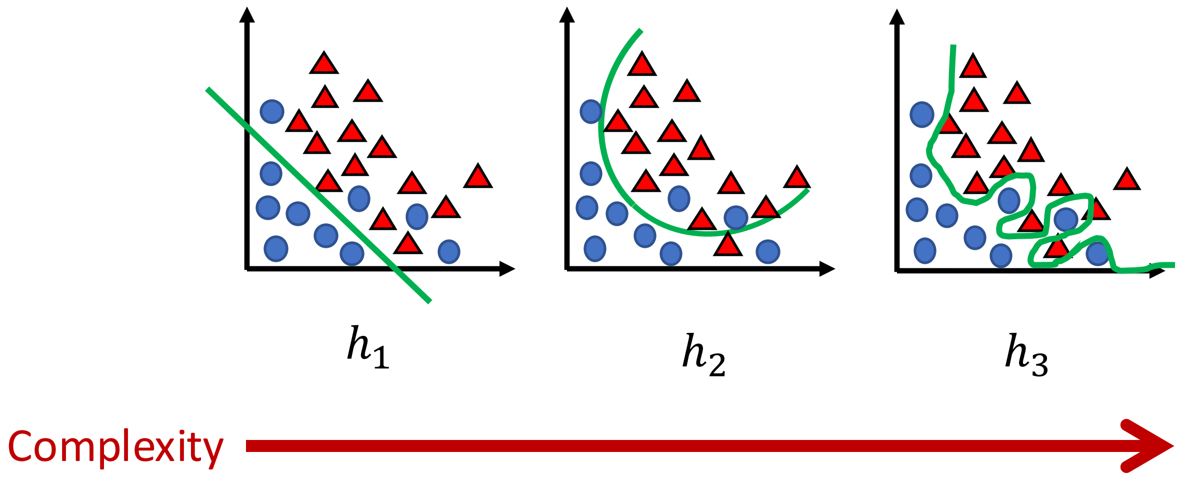
- Refers to the **size** and **expressiveness** of the hypothesis class.
- Indicates how intricate the relationships between input and output variables that the model can capture are.
- Higher model complexity allows for more sophisticated modeling of input-output relationships.
  - Example: polynomial regression model has a higher model complexity than linear regression model, thus it can model more complicated data.



Model Complexity: Logistic Regression

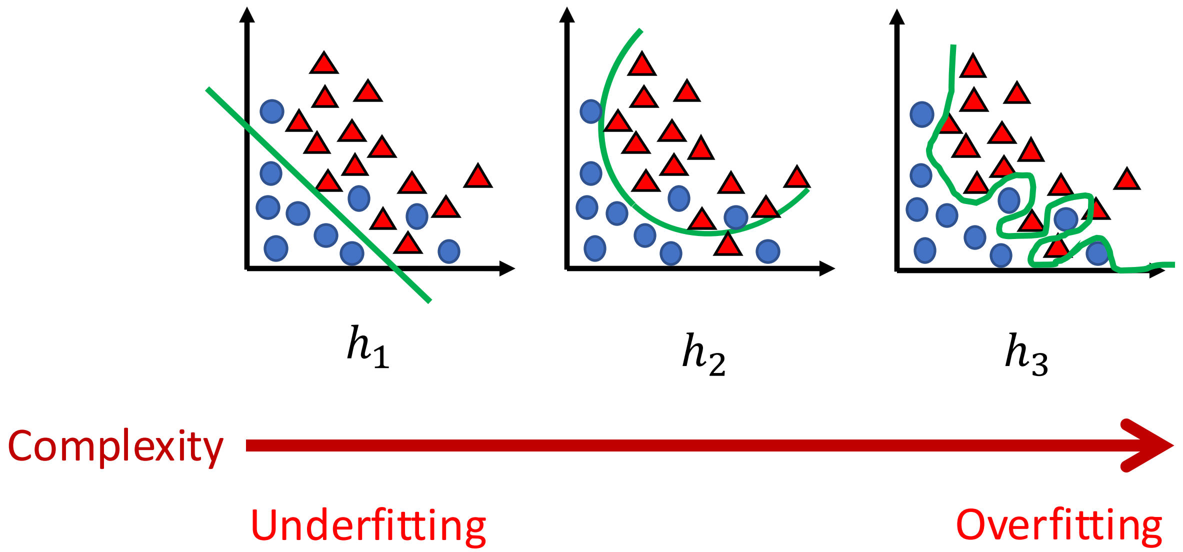
with Polynomial Features

- Low complexity $h_1(x) = \sigma(w_0 + w_1 x)$
- Medium complexity $h_2(x) = \sigma(w_0 + w_1 x + w_2 x^2)$
- High complexity $h_3(x) = \sigma(w_0 + w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4 + \cdots)$



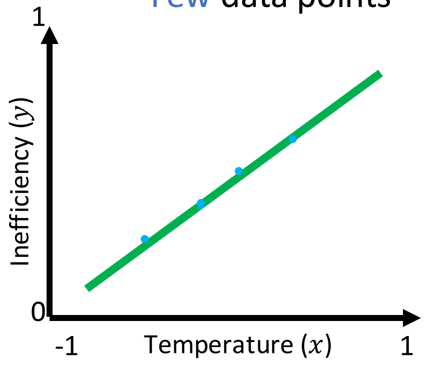
Model Complexity: Under- and Overfitting

- Low-complexity model <u>**cannot**</u> **capture the data**. Underfitting!
- Medium-complexity model **can capture the data**, with some exceptions.
- High-complexity model **may also capture the noise** of the data. Overfitting!

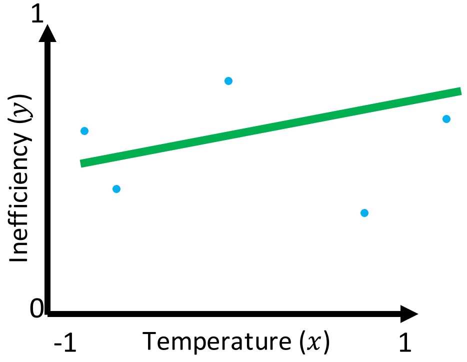


Case Study with Simple Model

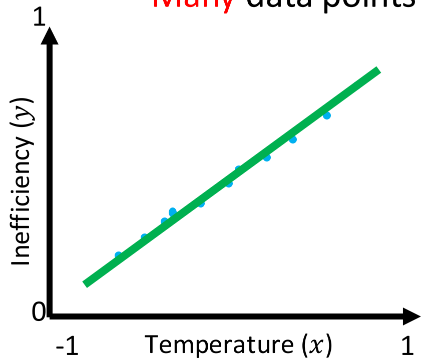
- Simple truth
- Few data points



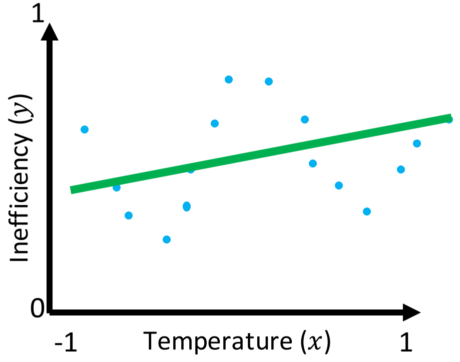
- Complex truth
- Few data points



- Simple truth
- Many data points

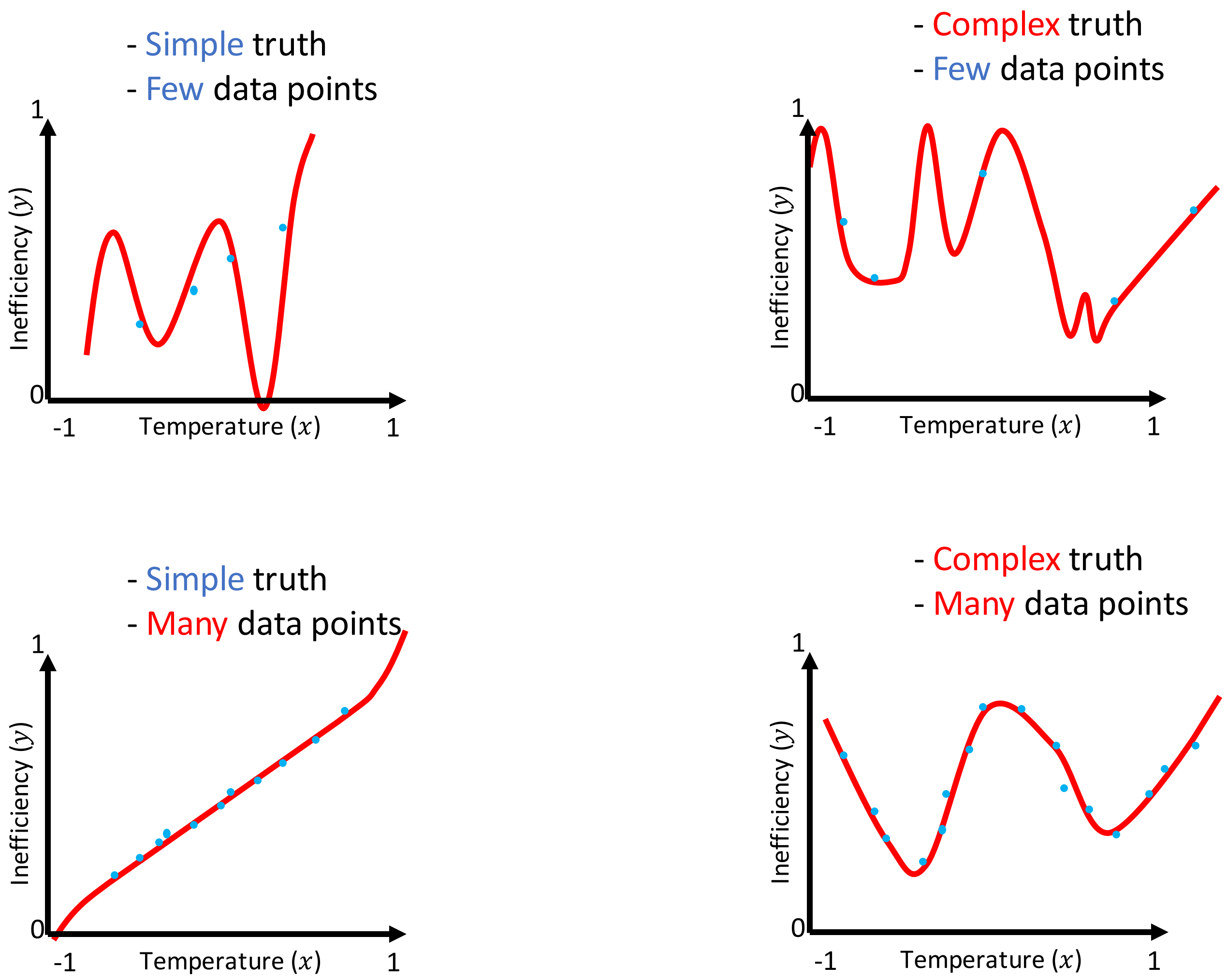


- Complex truth
- Many data points



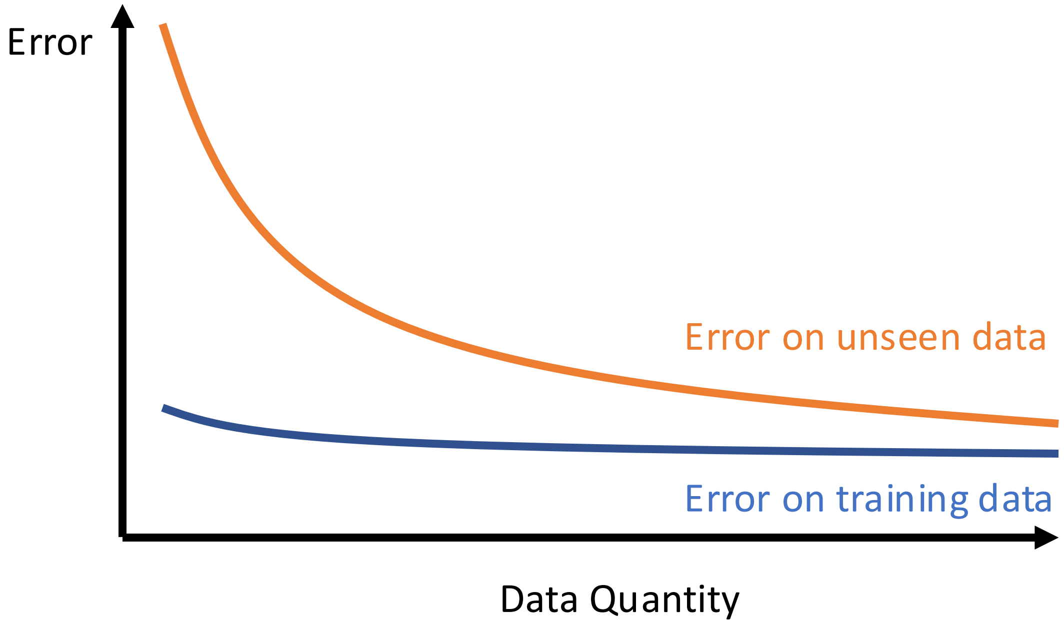
- Simple model is good for simple ground truth – More data points not necessary
- Simple model is bad for complex ground truth – More data points do not help – In ML, we say the model has a high bias.
- Retraining simple model with a different training set based on the same ground truth leads to essentially the same model – In ML, we say the model has a low variance.

Case Study with Complex Model

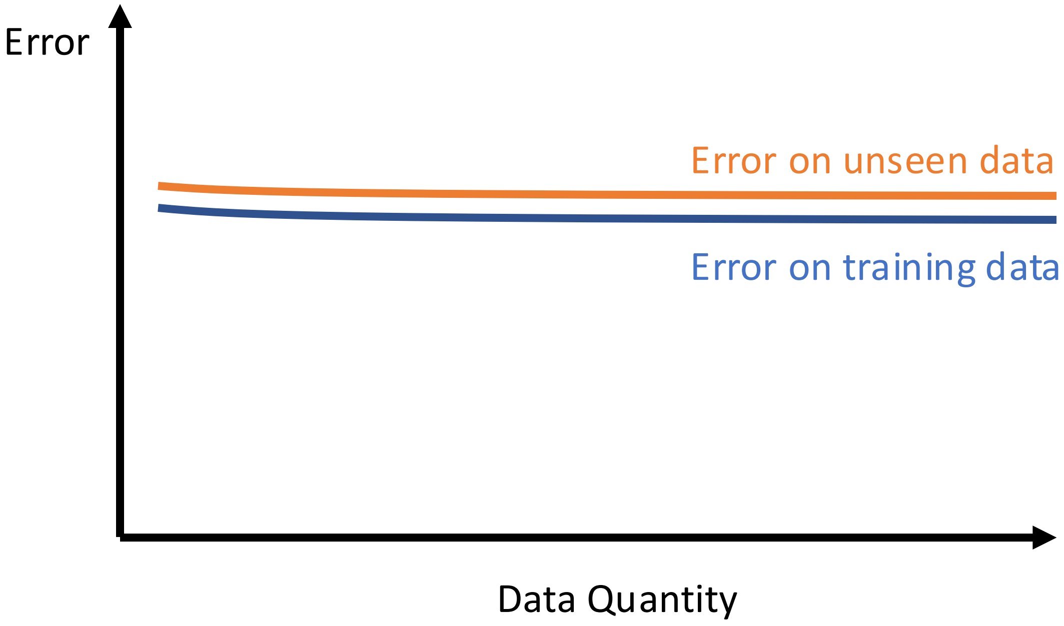


- Complex model overfits to few data points for both simple and complex group truth.
- Complex model can fit simple and complex ground truth if many data points are provided (and the noise is small) – In ML, we say the model has a low bias.
- Retraining a complex model with a different training set based on the same ground truth can lead to a very different model – In ML, we say the model has a high variance.

Model Complexity and Data Quantity vs Error



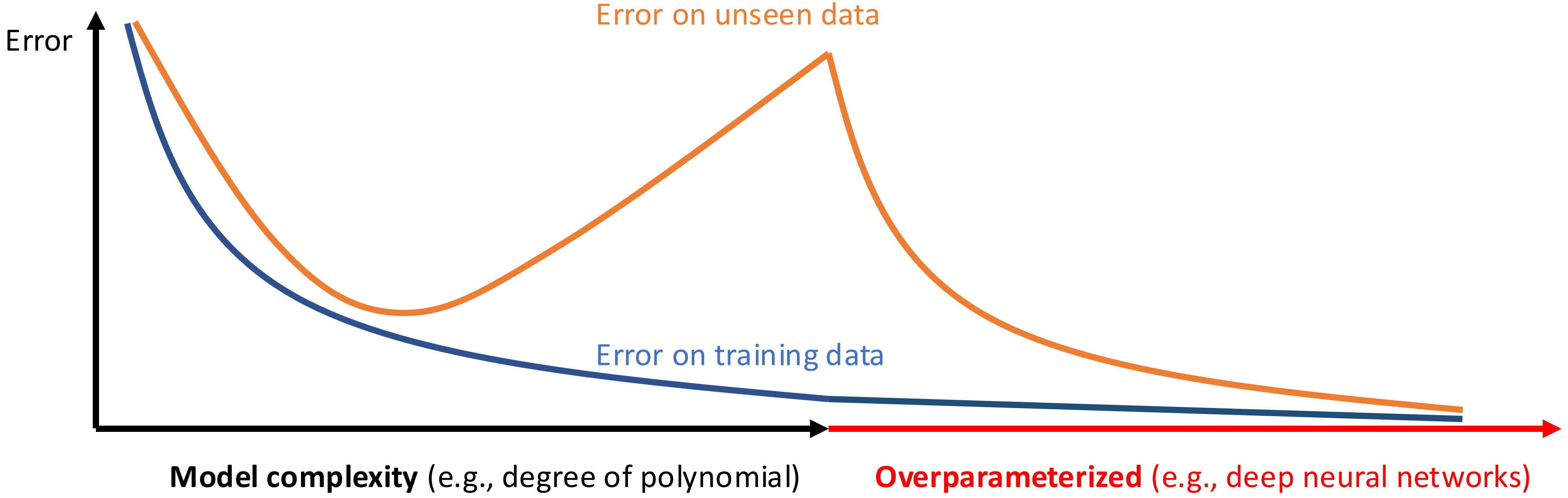
High-complexity Model



Low-complexity Model

Model Complexity vs Error

on not so large data



Hyperparameters

- **Hyperparameters** are settings that control the behavior of the training algorithm and model but are not learned from the data. They need to be set before the training process begins.
- Example:
  - Learning rate (e.g., 0.1, 0.5, …)
  - Feature transformations (e.g., polynomial degree)
  - Batch size and iterations in mini-batch gradient descent
- **Hyperparameters** vs. **Parameters**: Parameters are learned during training (e.g., weights in a linear model), while hyperparameters are predefined and adjusted manually (e.g., learning rate).

Hyperparameter Tuning

- **Hyperparameter tuning** is the process of optimizing the hyperparameters of a machine learning model to improve its performance. It is also known as **hyperparameter search**.
- **Objective**: The goal is to find the best combination of hyperparameters that maximize the model's performance.

Hyperparameter Tuning – Techniques

- **Grid search (exhaustive search)**
  - All possible combinations of a predefined set of hyperparameters are exhaustively tried.
  - **Example**: Trying all possible combinations of learning rates and regularization.
- **Random search**
  - Hyperparameters are randomly selected from a predefined distribution. Unlike grid search, it does not try every possible combination.
  - **Example**: Randomly sampling learning rates and regularization parameters.
- **Local search**
  - Use local search algorithms, such as hill-climbing, to iteratively optimize the hyperparameters. It starts with an initial set of hyperparameters and makes incremental changes to improve performance.
  - **Example**: Starting with an initial learning rate and regularization parameter and iteratively adjusting them to improve model performance using hill-climbing.
- **Successive halving**, **Bayesian optimization**, …

Many "off-the-shelves" packages available (e.g., in scikit-learn, hyperopt)!

Summary

- Logistic Regression: compute the probability of an input belonging to a class
  - Model: d dimensional input features: $h_w(x) = \sigma(\sum_{j=0}^{d} w_j x_j) = \sigma(w^T x)$
  - Binary Cross Entropy Loss:
    - $\frac{1}{N}\sum_{i=1}^{N} BCE\left(h_w\left(x^{(i)}\right), y^{(i)}\right)$, $BCE(y, \hat{y}) = -y\log(\hat{y}) - (1-y)\log(1-\hat{y})$
  - Non-linearly separable data: use feature transformations
- Learning via Gradient Descent: derivative is the same with linear regression!
- Multi-Class classification:
  - One-vs-one: create a classifier for each pair of classes, pick most votes
  - One-vs-rest: create a classifier for each class vs the rest of the classes, pick the highest output
- Advanced Topics in Supervised Learning
  - Generalization: performance on unseen data
  - Model Complexity: how expressive the model is, e.g., polynomial degree
  - Overfitting & Underfitting: fit too much to training data vs can’t fit even the training data
  - Hyperparameter Tuning: optimize the configuration of model and training

Further Reading (Optional)

- **History:** Historical use of logistic regression to investigate the root cause of the Challenger explosion in 1986.
  - (For example: https://cooperrc.github.io/data-driven-decisions/module_03/challenger.html)
- **Choosing the best hypothesis:** Model selection, cross-validation (AIAMA, Chapter 18.4)
- **Bias-Variance decomposition.** (PRML, Chapter 3.2)

Coming Up

- Next week: Recess Week, no tutorial
- Week 7: has Midterm, no tutorial
- Week 8: Lecture 7, has tutorial
- **Lecture 7**
  - Regularizations
  - Kernels
  - Support Vector Machines

To Do

- **Lecture Training 6**
  - +250 EXP
  - +100 Early bird bonus
- **Problem Set 3**
  - Will be released later today!