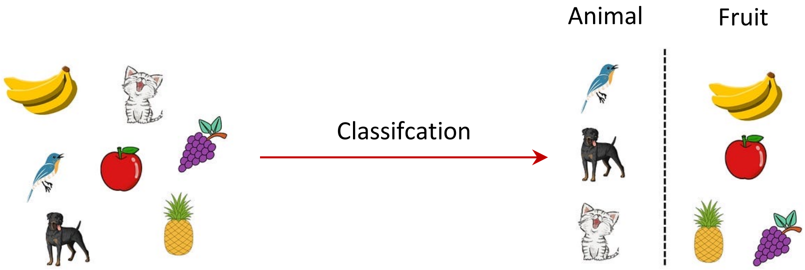
Logistic Regression
(Classification)

Classification

Definition of the word "Classification" from Oxford dictionary:

C1 [uncountable] the act or process of putting people or things into a group or class (= of classifying them)



Classification

In machine learning, a classification problem is a type of supervised learning task where the objective is to assign input data to one of several predefined categories or classes.

Examples of classification problems include:

- Binary Classification.
- Multi-class Classification.
- Multi-class Multi-label Classification.

Question: give examples for each type of classification problems.

Regression vs. Classification

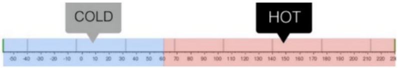
Regression

What will be the temperature tomorrow?

Fahrenheit

Classification

Will it be hot or cold tomorrow?



Fahrenheit

Comparison between Regression and Classification on weather prediction problem.

Binary classification

Binary classification is a type of classification task in machine learning where the objective is to categorize input data into one of **two classes**.

Examples of binary classification:

- Spam mail classification: *Spam / Non-spam*
- Online transaction fraud detetion: *Yes / No*
- Tumor recognition: *Malignant / Benign*

Label of binary classification: $y = \{0,1\}$

- 0 is negative class.
- 1 is possitive class.

Question: what does the label of multi-class classification look like?

Binary classification – Probablity calculation

Instead of just predicting the class, binary classifier gives the probability of the data sample being that class, for example:

- Spam mail classification: $P(\text{Spam}|\text{Email content})$
- Online transaction fraud detetion: $P(\text{Yes}|\text{Transaction details})$
- Tumor recognition: $P(\text{Malignant}|\text{Medical image})$

Recall that in binary classification:

- $0 \leq P(\text{event}) \leq 1$
- $P(\text{event}) + P(\neg\text{event}) = 1$

Therefore, to solve the binary classification problem, the classifier only needs to calculate $P(\text{event})$, then $P(\neg\text{event}) = 1 - P(\text{event})$.

Logistic regression

Logistic regression is a type of supervised learning algorithm used for classification tasks, particularly binary classification.

Logistic regression uses **sigmoid** function (or logistic function) to map the linear combination of input features to a value between 0 and 1.

The sigmoid function is defined as:

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

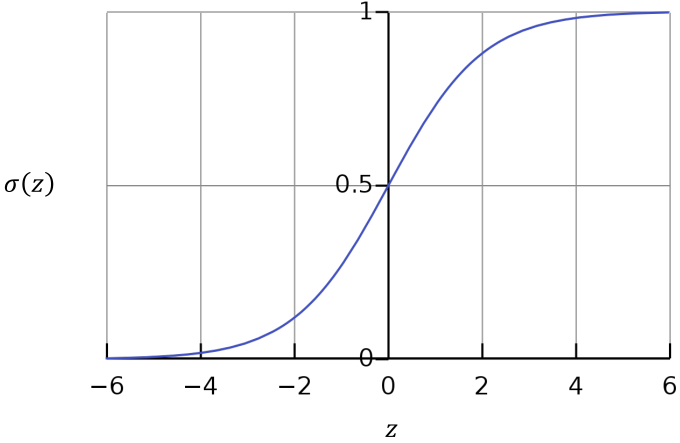
Here, z is the linear combination of input features, expressed as $z = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_d x_d$ where θ are the model parameters and x are input features.

Question: What is the motivation behind using z? Is it similar to a concept we've previously learned?

Logistic regression – Sigmoid function

Sigmoid function is to map input values to a range between 0 and 1.

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



Logistic regression – Output prediction

The output of the logistic function represents the probability of the input belonging to the positive class (e.g., class 1). Typically, a threshold (e.g., 0.5) is applied to decide the class label.

- If $\sigma(z) \geq 0.5$, the output is classified as class 1.
- If $\sigma(z) < 0.5$, the output is classified as class 0.

Example: spam mail classification with labels 1 (spam) and 0 (non-spam). Consider the threshold of 0.5 and a mail sample having $z = 2$:

$$\sigma(2) = \frac{1}{1 + e^{-2}} = 0.88 \geq 0.5$$

Therefore, this mail sample is classified as spam.

Question: What is the intuition and impact of having the threshold

- larger than 0.5?
- smaller than 0.5?

Logistic regression – Another way to look at it

Logistic regression is **linear regression wrapped in a sigmoid function**, ensuring that the output is a probability value between 0 and 1.

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

where $\theta^T x = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_d x_d$

Question: How can we train a logistic regression model?

Hints:

- Logistic regression is a supervised learning model.
- Recall how linear regression model was trained.

Logistic regression – Training

Question: How can we train a logistic regression model?

Hints:

- Logistic regression is a supervised learning model.
  - We have data features and data labels.
- Recall how linear regression model was trained.
  - Setup the model and the loss function.
  - Calculate gradient of the loss function.
  - Update model parameters using gradient descent.

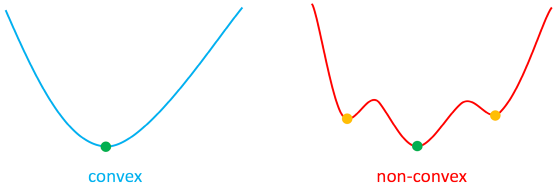
Logistic regression – Issue of MSE Loss function

Recall the MSE loss function of Linear regression

$$J(\theta) = \frac{1}{2n}\sum_{i=1}^{n}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$$

where $h_\theta(x) = \theta^T x$ which makes $J(\theta)$ a **quadratic** function (polynomial function of degree 2) and thus **convex**.

However, in Logistic regression $h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$, if we use the MSE loss function, $J(\theta)$ is a **non-convex** function that poses challenges for the convergence of the gradient descent algorithm.



Logistic regression – Binary Cross Entropy (BCE) Loss function

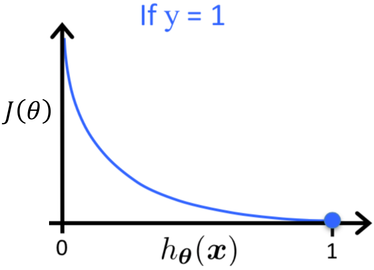
We want a convex loss function $J(\theta)$ of Logistic regression model as follows:

$$J(\theta) = \begin{cases} -\log\left(h_\theta(x)\right) & \text{if } y = 1 \\ -\log\left(1 - h_\theta(x)\right) & \text{if } y = 0 \end{cases}$$

Logistic regression – Intuition behind BCE Loss function

We want a convex loss function $J(\theta)$ of Logistic regression model as follows:

$$J(\theta) = \begin{cases} \boxed{-\log\big(h_\theta(x)\big) \quad \text{if } y = 1} \\ -\log\big(1 - h_\theta(x)\big) \quad \text{if } y = 0 \end{cases}$$



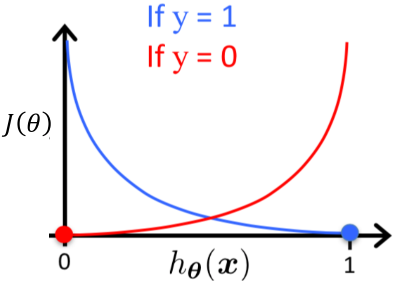
If $y = 1$:

- $J(\theta) = 0$ if prediction is correct.
- As $h_\theta(x) \to 0, J(\theta) \to \infty$
- Captures intuition that larger mistakes should get larger penalties, e.g., predict $h_\theta(x) = 0$, but $y = 1$.

Logistic regression – Intuition behind BCE Loss function

We want a convex loss function $J(\theta)$ of Logistic regression model as follows:

$$J(\theta) = \begin{cases} -\log\big(h_\theta(x)\big) & \text{if } y = 1 \\ \boxed{-\log\big(1 - h_\theta(x)\big) \quad \text{if } y = 0} \end{cases}$$



If $y = 0$:

- $J(\theta) = 0$ if prediction is correct.
- As $h_\theta(x) \to 1, J(\theta) \to \infty$
- Captures intuition that larger mistakes should get larger penalties, e.g., predict $h_\theta(x) = 1$, but $y = 0$.

Logistic regression – Simplification of BCE Loss function

We want a convex loss function $J(\theta)$ of Logistic regression model as follows:

$$J(\theta) = \begin{cases} -\log\big(h_\theta(x)\big) & \text{if } y = 1 \\ -\log\big(1 - h_\theta(x)\big) & \text{if } y = 0 \end{cases}$$

Question: How can the BCE Loss function be expressed in a single line?

Logistic regression – Simplification of BCE Loss function

We want a convex loss function $J(\theta)$ of Logistic regression model as follows:

$$J(\theta) = \begin{cases} -\log\big(h_\theta(x)\big) & \text{if } y = 1 \\ -\log\big(1 - h_\theta(x)\big) & \text{if } y = 0 \end{cases}$$

Question: How can the BCE Loss function be expressed in a single line?

Answer:

$$J(\theta) = -y\log\big(h_\theta(x)\big) - (1-y)\log\big(1 - h_\theta(x)\big)$$

- If $y = 1, J(\theta) = -\log\big(h_\theta(x)\big)$
- If $y = 0, J(\theta) = -\log\big(1 - h_\theta(x)\big)$

Logistic regression – BCE Loss function

The BCE loss function is averaged on all data samples:

$$J(\theta) = \frac{1}{n}\sum_{i=1}^{n}\left(-y^{(i)}\log\left(h_\theta\left(x^{(i)}\right)\right) - (1-y^{(i)})\log\left(1-h_\theta\left(x^{(i)}\right)\right)\right)$$

Objective:

$$\underset{\theta}{\text{minimize}}\, J(\theta)$$

To make prediction on a new data sample x':

$$h_{\theta^*}(x') = \frac{1}{1+e^{-\theta^* x'}} = P(y=1|x',\theta^*)$$

where θ^* is the value at which $J(\theta)$ is minimized, i.e., convergence point.

Logistic regression – Gradient descent algorithm

$$J(\theta) = \frac{1}{n}\sum_{i=1}^{n}\left(-y^{(i)}\log\left(h_\theta\left(x^{(i)}\right)\right) - (1-y^{(i)})\log\left(1-h_\theta\left(x^{(i)}\right)\right)\right)$$

To $\min\limits_{\theta} J(\theta)$, use Gradient descent algorithm:

- Initialize θ.
- Repeat until convergence:

$$\theta_j = \theta_j - \alpha\frac{\partial}{\partial\theta_j}J(\theta)$$

where $\frac{\partial}{\partial\theta_j}J(\theta) = \frac{1}{n}\sum_{i=1}^{n}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)x_j^{(i)}$

This looks identical to Linear regression. However, the form of the model is different: $h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$

Logistic regression – Regularization

$$J(\theta) = \frac{1}{n}\sum_{i=1}^{n}\left(-y^{(i)}\log\left(h_\theta\left(x^{(i)}\right)\right) - (1 - y^{(i)})\log\left(1 - h_\theta\left(x^{(i)}\right)\right)\right)$$

$$J_{regularized}(\theta) = J(\theta) + \underbrace{\frac{\lambda}{2}\sum_{j=1}^{d}\theta_j^2}_{\text{L2 regularization}}$$

To $\min_{\theta} J_{regularized}(\theta)$, use Gradient descent algorithm:

- Initialize θ.
- Repeat until convergence:

$$\theta_j = \theta_j - \alpha\frac{\partial}{\partial\theta_j}J_{regularized}(\theta)$$

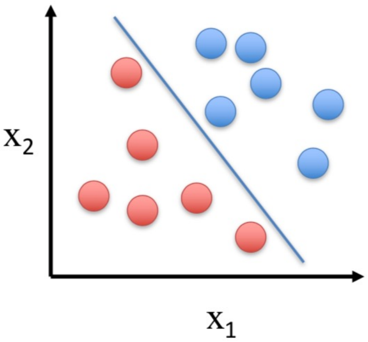
where $\frac{\partial}{\partial\theta_j}J_{regularized}(\theta) = \frac{1}{n}\sum_{i=1}^{n}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)x_j^{(i)} + \lambda\theta_j$

Multi-class classification

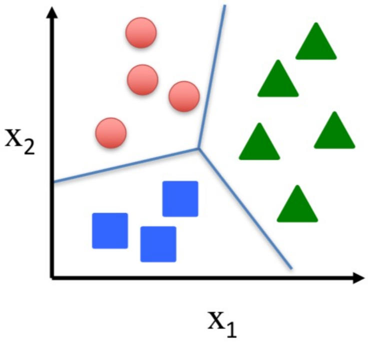
Example of multi-class classification:

- Object classification: desk / chair / monitor / keyboard.
- Animal classification: dog / cat / chicken.

Binary classification

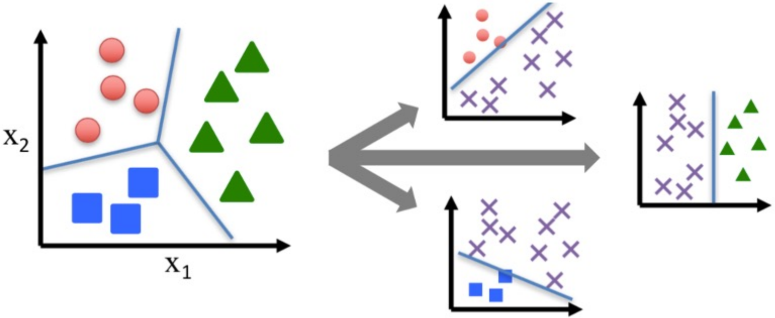


Multi-class classification



Question: how can Logistic regression be used to solve multi-class classification?

Multi-class classification – One-vs-all approach



Questions:

- How to determine the prediction output?
- What is the drawback of One-vs-all approach?

Multi-class classification – Softmax regression

Example: A multi-class classification includes dog, cat and chicken class. Given data features of a data sample, we want:

Data features

x_1

x_2

x_3

x_4

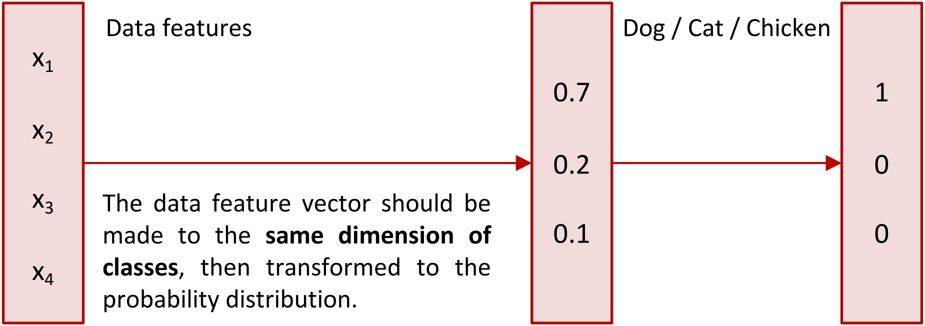
Dog / Cat / Chicken

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

But we better need a **probability distribution** over the classes, rather than direct predictions of the classes

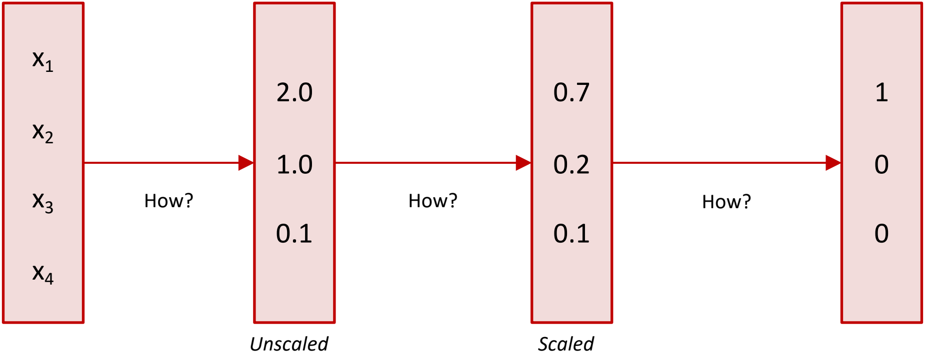
Multi-class classification – Softmax regression

Example: A multi-class classification includes dog, cat and chicken class. Given data features of a data sample, we want:



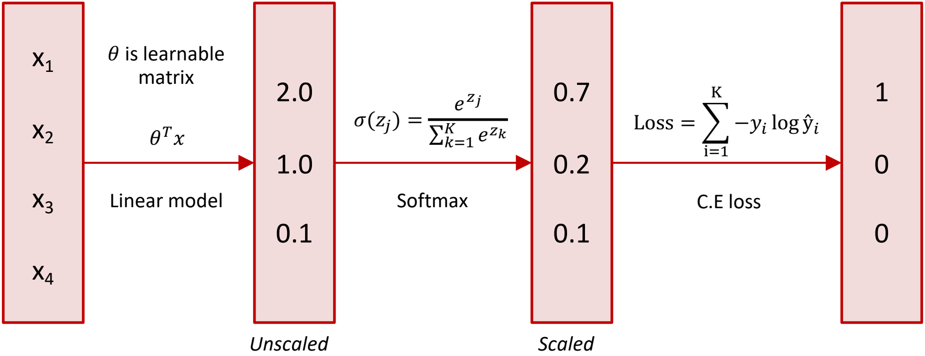
Multi-class classification – Softmax regression

Example: A multi-class classification includes dog, cat and chicken class. Given data features of a data sample, we want:



Multi-class classification – Softmax regression

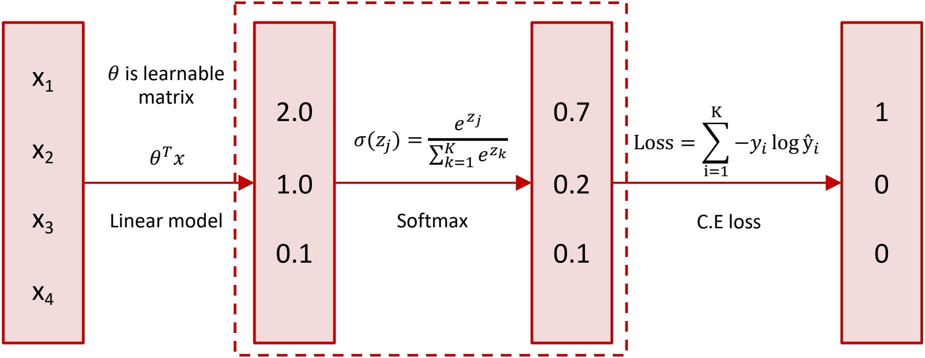
Softmax regression is a generalization of logistic regression to solve multi-class classification problems.



Note: In this example, the data feature vector has a shape of 1x4, the learnable matrix θ has a shape of 3x4, and the unscaled output vector has a shape of 1x3.

Softmax regression

Softmax regression is a generalization of logistic regression to solve multi-class classification problems.



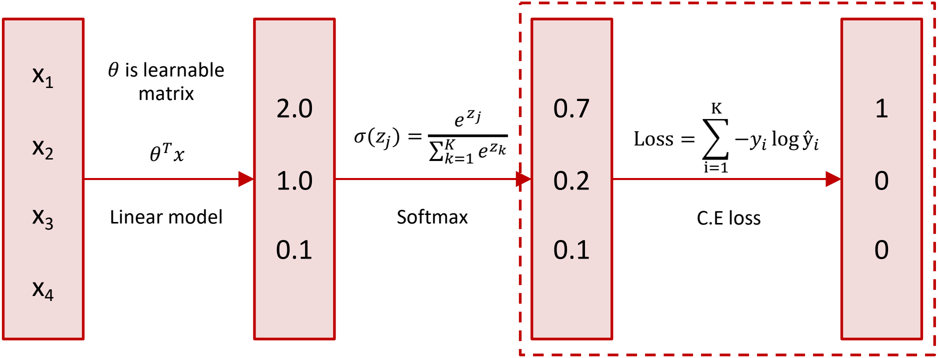
$$\sigma(2.0) = \frac{e^{2.0}}{e^{2.0} + e^{1.0} + e^{0.1}} = 0.7$$

$$\sigma(0.1) = \frac{e^{0.1}}{e^{2.0} + e^{1.0} + e^{0.1}} = 0.1$$

$$\sigma(1.0) = \frac{e^{1.0}}{e^{2.0} + e^{1.0} + e^{0.1}} = 0.2$$

Softmax regression

Softmax regression is a generalization of logistic regression to solve multi-class classification problems.



$$\text{Loss} = -1 \times log_2 0.7 - 0 \times log_2 0.2 - 0 \times log_2 0.1 = 0.51$$

This loss is then used to calculate the gradient and to update learnable matrix of parameters θ using the gradient descent algorithm.

Evaluation metrics

General method: calculate the difference between ground-truth labels and model predictions.

Example: testing 165 emails in a spam/non-spam classification problem.

	Prediction YES	**Prediction NO**
Actual YES	100	5
Actual NO	10	50

Confusion matrix

Evaluation metrics

Example: testing 165 emails in a spam/non-spam classification problem.

	Prediction YES	**Prediction NO**
Actual YES	100	5
Actual NO	10	50

Confusion matrix

- Precision = 100/(100+10) ~ 91%: how many predicted items are relevant.
- Recall = 100/(100+5) ~ 95%: how many relevant items are predicted.

Evaluation metrics

Example: testing 165 emails in a spam/non-spam classification problem.

	Prediction YES	**Prediction NO**
Actual YES	True Positive TP	False Negative FN
Actual NO	False Positive FP	True Negative TN

Confusion matrix

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

Summary

Binary Classification

Decision Boundary

Logistic Regression

- Sigmoid function
- Cost Function
- Optimization
- Regularization

Multi-class (Multinomial Classification)

- One-vs-all
- Softmax regression

Evaluation metrics

Q&A

Thank you