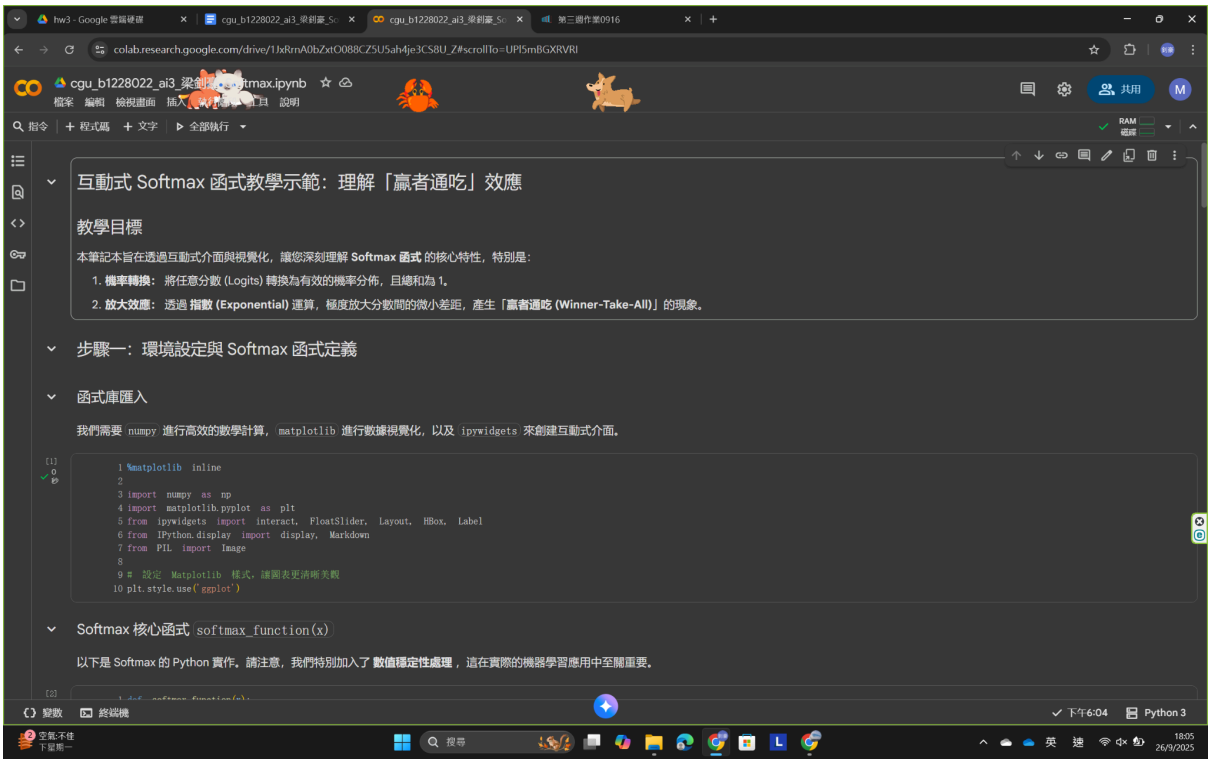
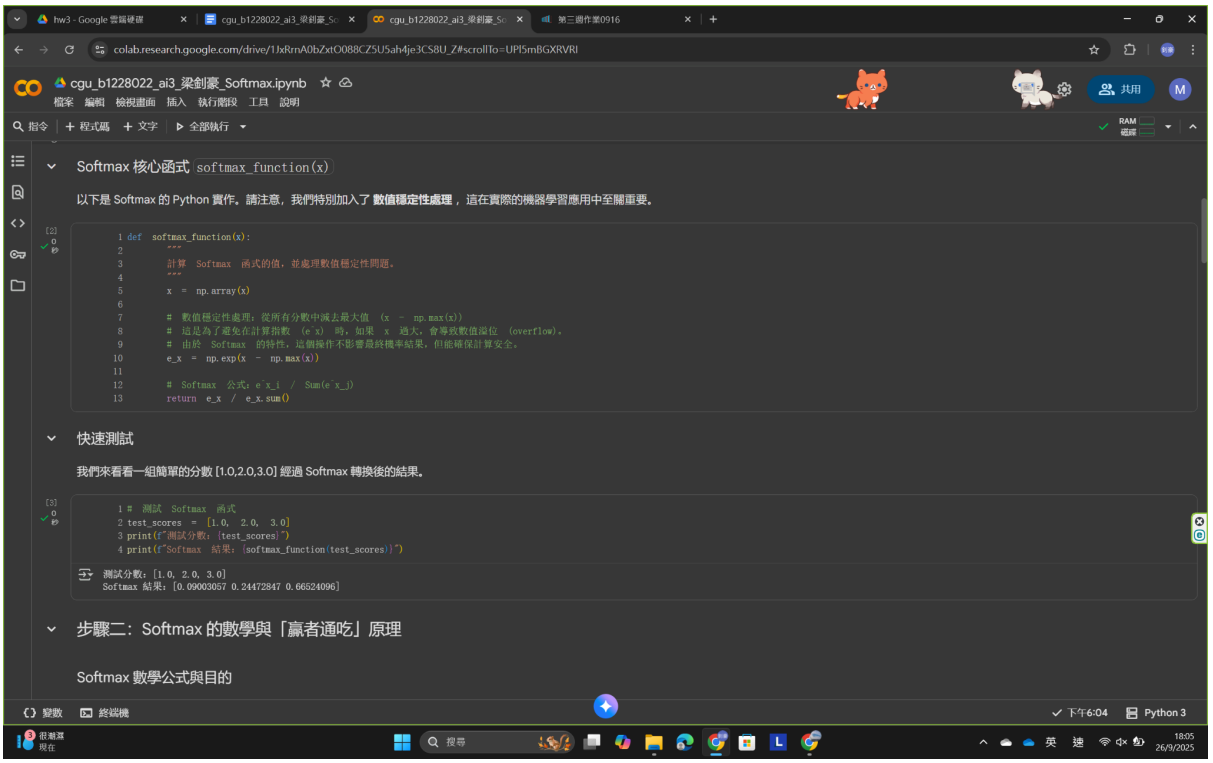
學校:長庚大學　　系級:人工智慧學系 大三

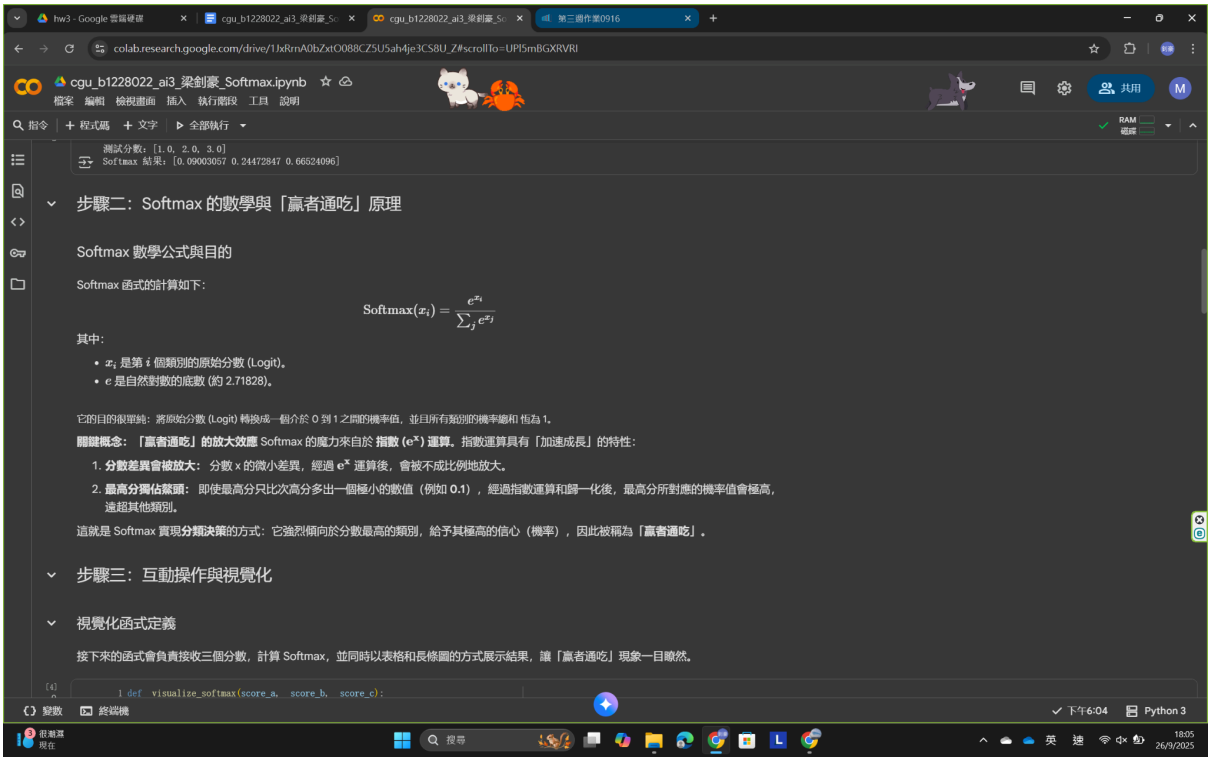
學號:B1228022　　姓名:梁劍豪　主題:互動式 Softmax

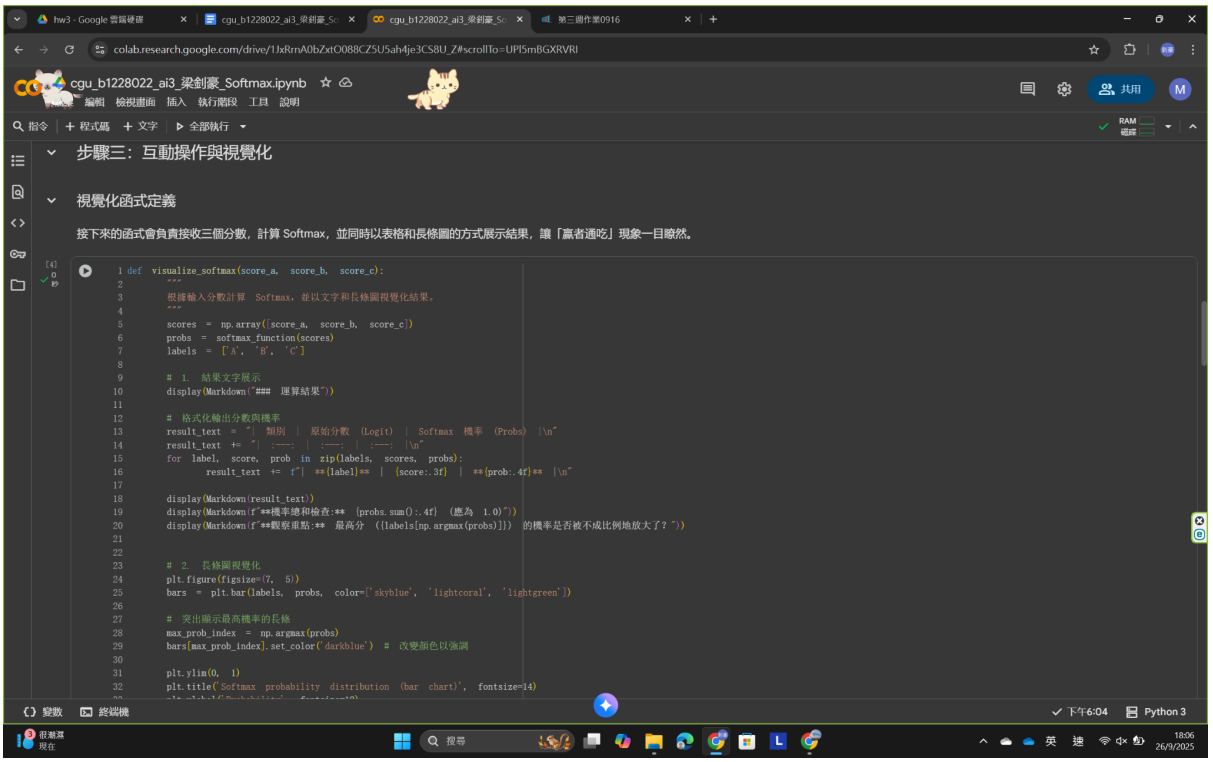
Colab:

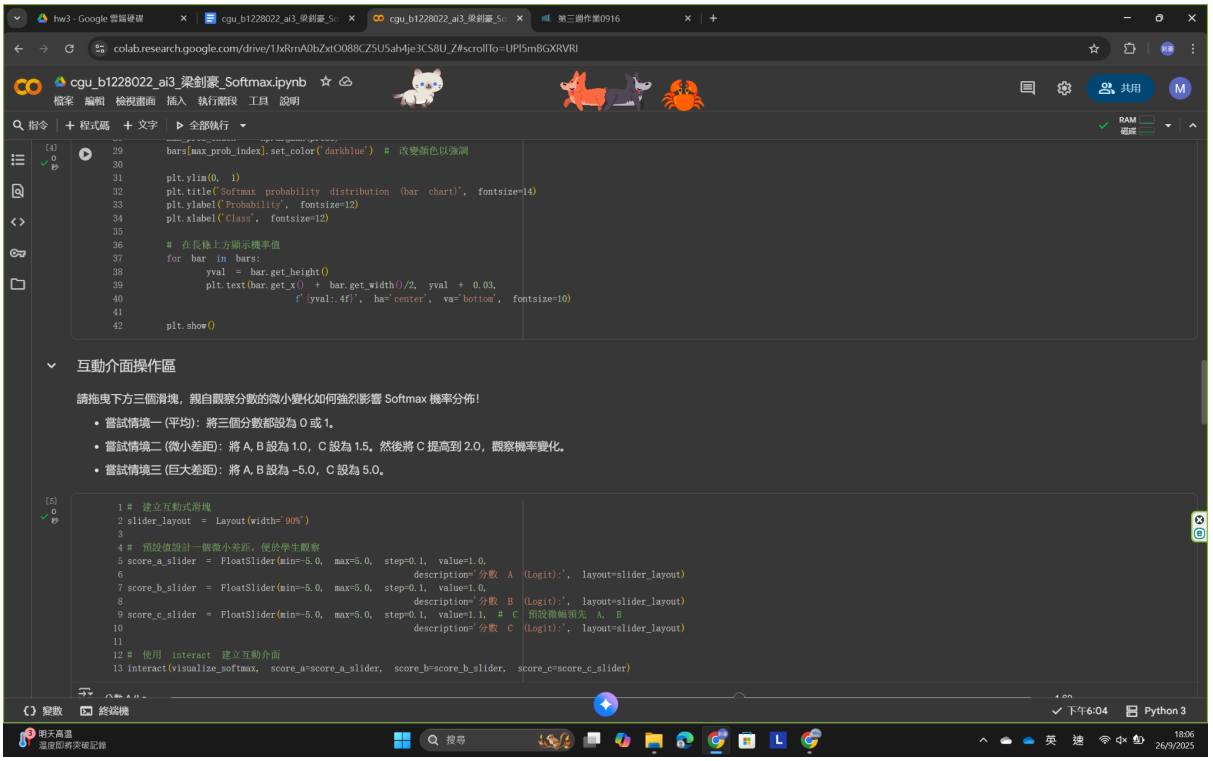
https://colab.research.google.com/drive/1JxRrnA0bZxtO088CZ5U5ah4je3CS8U_Z?usp=sharing

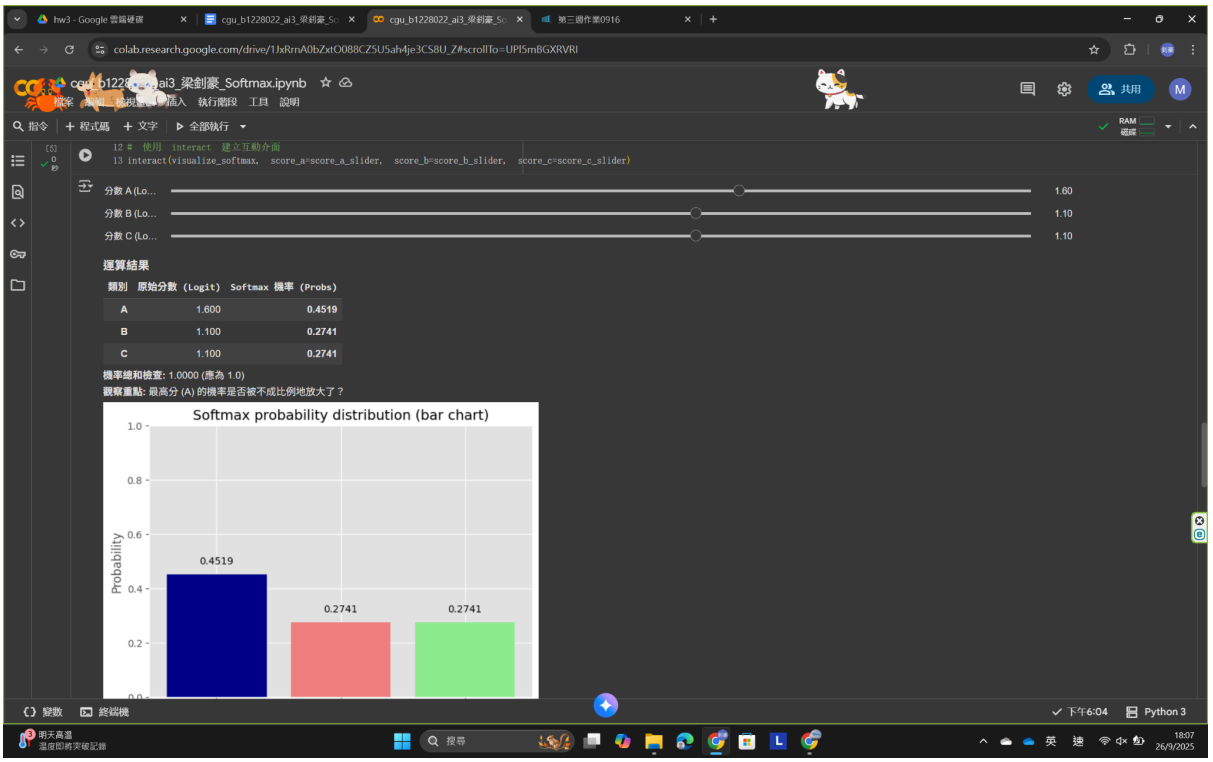


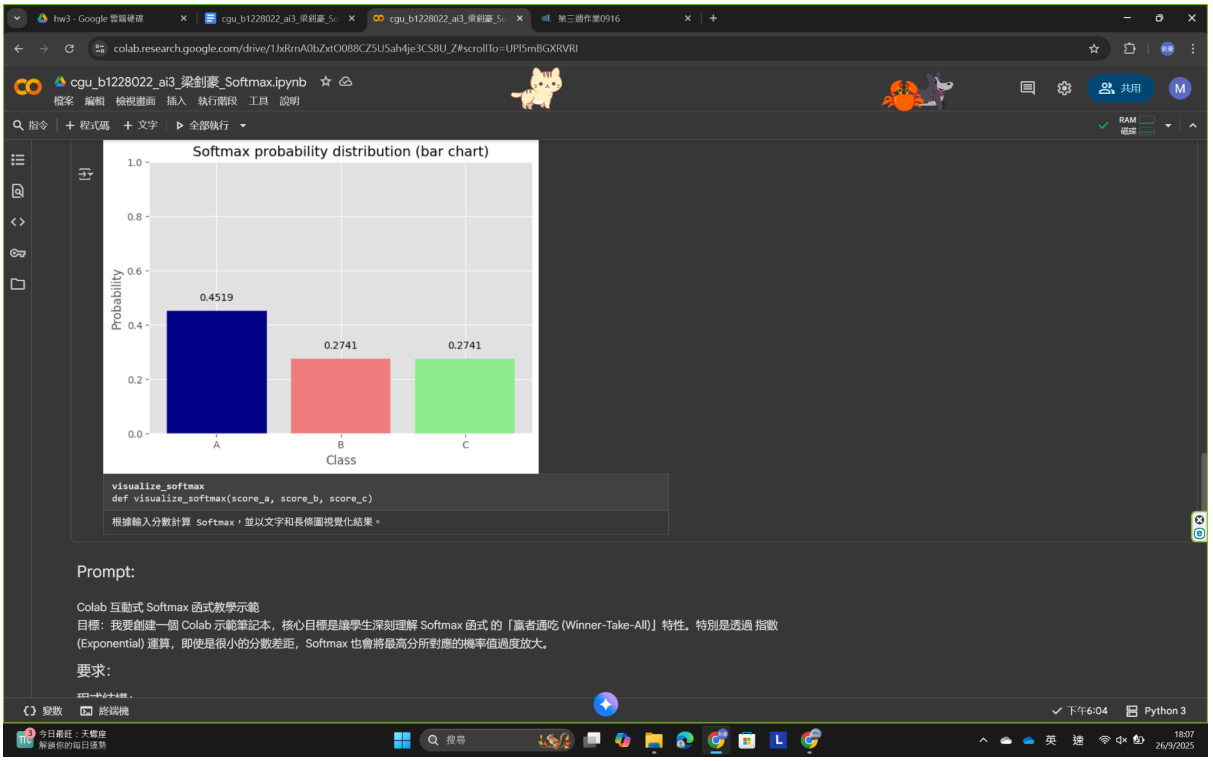



hw3 - Google 雲端硬碟
cgu_b1228022_ai3_梁劍豪_Softmax.ipynb
colab.research.google.com/drive/1JxRrnA0bZxtO088CZ5U5ah4je3CS8U_Z#scrollTo=UPl5mBGXRVRI
cgu_b1228022_ai3_梁劍豪_Softmax.ipynb
檔案 編輯 檢視畫面 插入 執行階段 工具 說明
共用
測試分數：[1.0, 2.0, 3.0]
Softmax 結果：[0.09003057 0.24472847 0.66524096]
步驟二：Softmax 的數學與「贏者通吃」原理
Softmax 數學公式與目的
Softmax 函式的計算如下：
$$\text{Softmax}(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$$
其中：
x_i 是第 i 個類別的原始分數 (Logit)。
e 是自然對數的底數 (約 2.71828)。
它的目的很單純：將原始分數 (Logit) 轉換成一個介於 0 到 1 之間的機率值，並且所有類別的機率總和 恆為 1。
關鍵概念：「贏者通吃」的放大效應 Softmax 的魔力來自於 指數 (e^x) 運算。指數運算具有「加速成長」的特性：
1. 分數差異會被放大： 分數 x 的微小差異，經過 e^x 運算後，會被不成比例地放大。
2. 最高分獨佔鰲頭： 即使最高分只比次高分多出一個極小的數值（例如 0.1），經過指數運算和歸一化後，最高分所對應的機率值會極高，遠超其他類別。
這就是 Softmax 實現分類決策的方式：它強烈傾向於分數最高的類別，給予其極高的信心（機率），因此被稱為「贏者通吃」。
步驟三：互動操作與視覺化
視覺化函式定義
接下來的函式會負責接收三個分數，計算 Softmax，並同時以表格和長條圖的方式展示結果，讓「贏者通吃」現象一目瞭然。
[4]
1 def visualize_softmax(score_a, score_b, score_c):
變數 終端機
下午6:04 Python 3
搜尋
18:05
26/9/2025


hw3 - Google 雲端硬碟
cgu_b1228022_ai3_梁劍豪_Softmax.ipynb
colab.research.google.com/drive/1JxRrnA0bZxtO088CZ5U5ah4je3CS8U_Z#scrollTo=UPl5mBGXRVRI
cgu_b1228022_ai3_梁劍豪_Softmax.ipynb
檔案 編輯 檢視畫面 插入 執行階段 工具 說明
共用
步驟三：互動操作與視覺化
視覺化函式定義
接下來的函式會負責接收三個分數，計算 Softmax，並同時以表格和長條圖的方式展示結果，讓「贏者通吃」現象一目瞭然。
[4]
def visualize_softmax(score_a, score_b, score_c):
"""
根據輸入分數計算 Softmax，並以文字和長條圖視覺化結果。
"""
scores = np.array([score_a, score_b, score_c])
probs = softmax_function(scores)
labels = ['A', 'B', 'C']
# 1. 結果文字展示
display(Markdown("### 運算結果"))
# 格式化輸出分數與機率
result_text = "| 類別 | 原始分數 (Logit) | Softmax 機率 (Probs) |\n"
result_text += "| :---: | :---: | :---: |\n"
for label, score, prob in zip(labels, scores, probs):
result_text += f"| **{label}** | {score:.3f} | **{prob:.4f}** |\n"
display(Markdown(result_text))
display(Markdown(f"**機率總和檢查:** {probs.sum():.4f} (應為 1.0)"))
display(Markdown(f"**觀察重點:** 最高分 ({labels[np.argmax(probs)]}) 的機率是否被不成比例地放大了？"))
# 2. 長條圖視覺化
plt.figure(figsize=(7, 5))
bars = plt.bar(labels, probs, color=['skyblue', 'lightcoral', 'lightgreen'])
# 突出顯示最高機率的長條
max_prob_index = np.argmax(probs)
bars[max_prob_index].set_color('darkblue') # 改變顏色以強調
plt.ylim(0, 1)
plt.title('Softmax probability distribution (bar chart)', fontsize=14)
變數 終端機
下午6:04 Python 3
搜尋
18:06
26/9/2025


hw3 - Google 雲端硬碟
cgu_b1228022_ai3_梁劍豪_Softmax.ipynb
colab.research.google.com/drive/1JxRrnA0bZxtO088CZ5U5ah4je3CS8U_Z#scrollTo=UPl5mBGXRVRI
檔案 編輯 檢視畫面 插入 執行階段 工具 說明
共用
指令 程式碼 文字 全部執行
bars[max_prob_index].set_color('darkblue') # 改變顏色以強調
plt.ylim(0, 1)
plt.title('Softmax probability distribution (bar chart)', fontsize=14)
plt.ylabel('Probability', fontsize=12)
plt.xlabel('Class', fontsize=12)
# 在長條上方顯示機率值
for bar in bars:
yval = bar.get_height()
plt.text(bar.get_x() + bar.get_width()/2, yval + 0.03,
f'{yval:.4f}', ha='center', va='bottom', fontsize=10)
plt.show()
互動介面操作區
請拖曳下方三個滑塊，親自觀察分數的微小變化如何強烈影響 Softmax 機率分佈！
• 嘗試情境一 (平均)：將三個分數都設為 0 或 1。
• 嘗試情境二 (微小差距)：將 A, B 設為 1.0，C 設為 1.5。然後將 C 提高到 2.0，觀察機率變化。
• 嘗試情境三 (巨大差距)：將 A, B 設為 -5.0，C 設為 5.0。
# 建立互動式滑塊
slider_layout = Layout(width='90%')
# 預設值設計一個微小差距，便於學生觀察
score_a_slider = FloatSlider(min=-5.0, max=5.0, step=0.1, value=1.0,
description='分數 A (Logit):', layout=slider_layout)
score_b_slider = FloatSlider(min=-5.0, max=5.0, step=0.1, value=1.0,
description='分數 B (Logit):', layout=slider_layout)
score_c_slider = FloatSlider(min=-5.0, max=5.0, step=0.1, value=1.1, # C 預設微幅領先 A, B
description='分數 C (Logit):', layout=slider_layout)
# 使用 interact 建立互動介面
interact(visualize_softmax, score_a=score_a_slider, score_b=score_b_slider, score_c=score_c_slider)
變數 終端機
下午6:04 Python 3
明天高溫 溫度即將突破記錄
搜尋
英 速
18:06 26/9/2025


hw3 - Google 雲端硬碟
cgu_b1228022_ai3_梁劍豪_Softmax.ipynb
colab.research.google.com/drive/1JxRrnA0bZxtO088CZ5U5ah4je3CS8U_Z#scrollTo=UPl5mBGXRVRI
檔案 編輯 檢視畫面 插入 執行階段 工具 說明
共用
指令 程式碼 文字 全部執行
# 使用 interact 建立互動介面
interact(visualize_softmax, score_a=score_a_slider, score_b=score_b_slider, score_c=score_c_slider)
分數 A (Lo... 1.60
分數 B (Lo... 1.10
分數 C (Lo... 1.10
運算結果
類別 原始分數 (Logit) Softmax 機率 (Probs)
A 1.600 0.4519
B 1.100 0.2741
C 1.100 0.2741
機率總和檢查: 1.0000 (應為 1.0)
觀察重點: 最高分 (A) 的機率是否被不成比例地放大了？
Softmax probability distribution (bar chart)
Probability
1.0
0.8
0.6
0.4
0.2
0.0
0.4519
0.2741
0.2741
變數 終端機
下午6:04 Python 3
明天高溫 溫度即將突破記錄
搜尋
英 速
18:07 26/9/2025




```
visualize_softmax
def visualize_softmax(score_a, score_b, score_c)
```

根據輸入分數計算 Softmax，並以文字和長條圖視覺化結果。

Prompt:

Colab 互動式 Softmax 函式教學示範
目標：我要創建一個 Colab 示範筆記本，核心目標是讓學生深刻理解 Softmax 函式 的「贏者通吃 (Winner-Take-All)」特性。特別是透過 指數 (Exponential) 運算，即使是很小的分數差距，Softmax 也會將最高分所對應的機率值過度放大。

要求：

程式結構：

請將程式碼分段編寫，並在每個程式碼儲存格前，加入清晰、教育性強的 Markdown 說明儲存格。

核心函式：

編寫一個名為 softmax_function(x) 的 Python 函式，它接受一個 Numpy 陣列 (array) 或 列表 (list) 作為輸入，並回傳其 Softmax 結果。請確保在計算中處理數值穩定性（例如，減去 max(x)）。

互動介面：

使用 ipywidgets 庫中的 interact 函式，建立一個互動式介面。
學生可以輸入 三個數字 (例如，score_a, score_b, score_c) 作為模型的輸出分數 (logits)。
介面應即時顯示這三個分數經過 softmax_function 處理後的機率值。
數據展示：

結果展示應包含：

原始輸入分數。
Softmax 計算後的機率值。
一個簡潔的 長條圖 (Bar Chart) 來視覺化機率分佈，強調最高機率的「獨大」現象。
教學重點：在 Markdown 說明中，明確闡述以下概念：
Softmax 的數學公式和用途（將分數轉換為機率分佈）。
指數運算對分數差異的放大效應（即「贏者通吃」）。
機率總和為 1 的特性。

最終產出：

一個可直接複製到 Colab 中運行的 Markdown/Python 混合 筆記本內容。

Prompt:

Colab 互動式 Softmax 函式教學示範
目標：我要創建一個 Colab 示範筆記本，核心目標是讓學生深刻理解 Softmax 函式 的「贏者通吃 (Winner-Take-All)」特性。特別是透過 指數 (Exponential) 運算，即使是很小的分數差距，Softmax 也會將最高分所對應的機率值過度放大。

要求：

程式結構：

請將程式碼分段編寫，並在每個程式碼儲存格前，加入清晰、教育性強的 Markdown 說明儲存格。

核心函式：

編寫一個名為 softmax_function(x) 的 Python 函式，它接受一個 Numpy 陣列 (array) 或 列表 (list) 作為輸入，並回傳其 Softmax 結果。請確保在計算中處理數值穩定性（例如，減去 max(x)）。

互動介面：

使用 ipywidgets 庫中的 interact 函式，建立一個互動式介面。
學生可以輸入 三個數字 (例如，score_a, score_b, score_c) 作為模型的輸出分數 (logits)。
介面應即時顯示這三個分數經過 softmax_function 處理後的機率值。
數據展示：

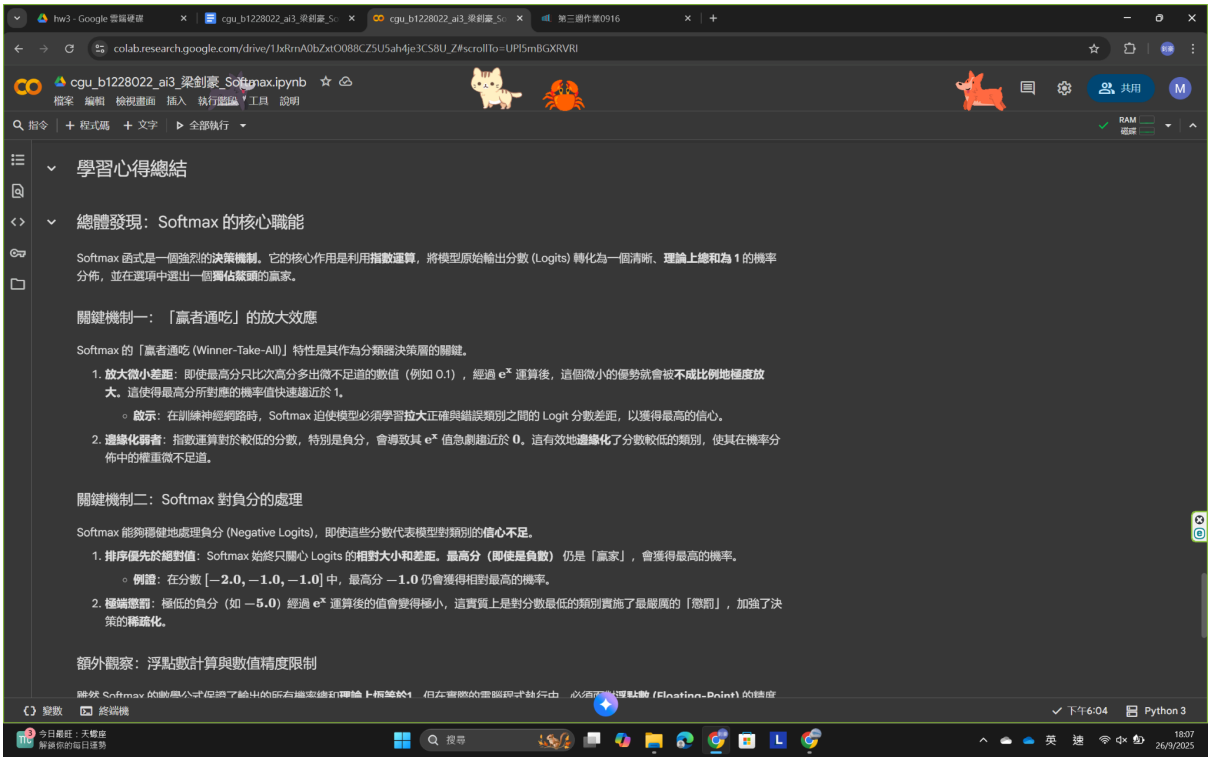
結果展示應包含：

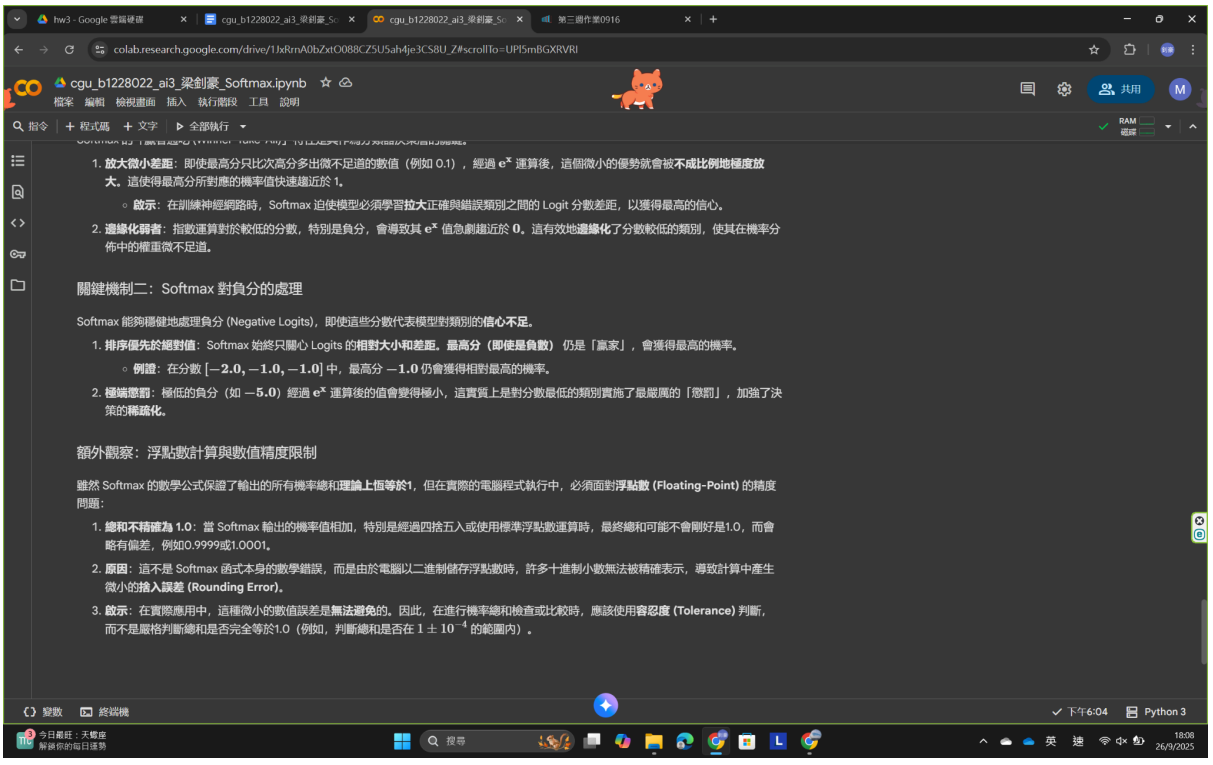
原始輸入分數。
Softmax 計算後的機率值。
一個簡潔的 長條圖 (Bar Chart) 來視覺化機率分佈，強調最高機率的「獨大」現象。
教學重點：在 Markdown 說明中，明確闡述以下概念：
Softmax 的數學公式和用途（將分數轉換為機率分佈）。
指數運算對分數差異的放大效應（即「贏者通吃」）。
機率總和為 1 的特性。

最終產出：

一個可直接複製到 Colab 中運行的 Markdown/Python 混合 筆記本內容。

心得


colab.research.google.com/drive/1JxRrnA0bZxtO088CZ5U5ah4je3CS8U_Z#scrollTo=UPl5mBGXRVRI
cgu_b1228022_ai3_梁劍豪_Softmax.ipynb
檔案 編輯 檢視畫面 插入 執行階段 工具 說明
共用
學習心得總結
總體發現：Softmax 的核心職能
Softmax 函式是一個強烈的決策機制。它的核心作用是利用指數運算，將模型原始輸出分數 (Logits) 轉化為一個清晰、理論上總和為 1 的機率分佈，並在選項中選出一個獨佔鰲頭的贏家。
關鍵機制一：「贏者通吃」的放大效應
Softmax 的「贏者通吃 (Winner-Take-All)」特性是其作為分類器決策層的關鍵。
1. 放大微小差距：即使最高分只比次高分多出微不足道的數值（例如 0.1），經過 e^x 運算後，這個微小的優勢就會被不成比例地極度放大。這使得最高分所對應的機率值快速趨近於 1。
◦ 啟示：在訓練神經網路時，Softmax 迫使模型必須學習拉大正確與錯誤類別之間的 Logit 分數差距，以獲得最高的信心。
2. 邊緣化弱者：指數運算對於較低的分數，特別是負分，會導致其 e^x 值急劇趨近於 0。這有效地邊緣化了分數較低的類別，使其在機率分佈中的權重微不足道。
關鍵機制二：Softmax 對負分的處理
Softmax 能夠穩健地處理負分 (Negative Logits)，即使這些分數代表模型對類別的信心不足。
1. 排序優先於絕對值：Softmax 始終只關心 Logits 的相對大小和差距。最高分（即使是負數）仍是「贏家」，會獲得最高的機率。
◦ 例證：在分數 $[-2.0, -1.0, -1.0]$ 中，最高分 -1.0 仍會獲得相對最高的機率。
2. 極端懲罰：極低的負分（如 -5.0）經過 e^x 運算後的值會變得極小，這實質上是對分數最低的類別實施了最嚴厲的「懲罰」，加強了決策的稀疏化。
額外觀察：浮點數計算與數值精度限制
雖然 Softmax 的數學公式保證了輸出的所有機率總和理論上恆等於1，但在實際的電腦程式執行中，必須面對浮點數 (Floating-Point) 的精度
變數 終端機
下午6:04 Python 3
18:07
26/9/2025


colab.research.google.com/drive/1JxRrnA0bZxtO088CZ5U5ah4je3CS8U_Z#scrollTo=UPl5mBGXRVRI
cgu_b1228022_ai3_梁劍豪_Softmax.ipynb
檔案 編輯 檢視畫面 插入 執行階段 工具 說明
共用
1. 放大微小差距：即使最高分只比次高分多出微不足道的數值（例如 0.1），經過 e^x 運算後，這個微小的優勢就會被不成比例地極度放大。這使得最高分所對應的機率值快速趨近於 1。
◦ 啟示：在訓練神經網路時，Softmax 迫使模型必須學習拉大正確與錯誤類別之間的 Logit 分數差距，以獲得最高的信心。
2. 邊緣化弱者：指數運算對於較低的分數，特別是負分，會導致其 e^x 值急劇趨近於 0。這有效地邊緣化了分數較低的類別，使其在機率分佈中的權重微不足道。
關鍵機制二：Softmax 對負分的處理
Softmax 能夠穩健地處理負分 (Negative Logits)，即使這些分數代表模型對類別的信心不足。
1. 排序優先於絕對值：Softmax 始終只關心 Logits 的相對大小和差距。最高分（即使是負數）仍是「贏家」，會獲得最高的機率。
◦ 例證：在分數 $[-2.0, -1.0, -1.0]$ 中，最高分 -1.0 仍會獲得相對最高的機率。
2. 極端懲罰：極低的負分（如 -5.0）經過 e^x 運算後的值會變得極小，這實質上是對分數最低的類別實施了最嚴厲的「懲罰」，加強了決策的稀疏化。
額外觀察：浮點數計算與數值精度限制
雖然 Softmax 的數學公式保證了輸出的所有機率總和理論上恆等於1，但在實際的電腦程式執行中，必須面對浮點數 (Floating-Point) 的精度問題：
1. 總和不精確為 1.0：當 Softmax 輸出的機率值相加，特別是經過四捨五入或使用標準浮點數運算時，最終總和可能不會剛好是1.0，而會略有偏差，例如0.9999或1.0001。
2. 原因：這不是 Softmax 函式本身的數學錯誤，而是由於電腦以二進制儲存浮點數時，許多十進制小數無法被精確表示，導致計算中產生微小的捨入誤差 (Rounding Error)。
3. 啟示：在實際應用中，這種微小的數值誤差是無法避免的。因此，在進行機率總和檢查或比較時，應該使用容忍度 (Tolerance) 判斷，而不是嚴格判斷總和是否完全等於1.0（例如，判斷總和是否在 1 ± 10^{-4} 的範圍內）。
變數 終端機
下午6:04 Python 3
18:08
26/9/2025