variety_calc

February 11, 2025

[1]:
```
'''
"title": "variety_calc",
"author": "sunnyxxy"
'''

import numpy as np
import matplotlib.pyplot as plt

def rao_quadratic_entropy_log(values, log_constant=1):
    values = np.array(values)

    # Determine the unique categories and their counts
    unique, counts = np.unique(values, return_counts=True)
    p = counts / counts.sum()  # relative frequencies

    # log metrics
    distance_func = lambda x, y: np.log(log_constant + abs(x - y))

    # Compute the distance (dissimilarity) matrix for the unique values
    n = len(unique)
    dist_matrix = np.zeros((n, n))
    for i in range(n):
        for j in range(n):
            dist_matrix[i, j] = distance_func(unique[i], unique[j])

    # Compute Rao's Quadratic Entropy: Q = sum_{i,j} p_i * p_j * d(i, j)
    Q = np.sum(np.outer(p, p) * dist_matrix)
    return Q

def variety(note_seq): # assume that note_seq already is sorted by head
    heads = [n[1] for n in note_seq]
    tails = [n[2] for n in note_seq] # -1 for rice is included
    tails.sort()
    head_gaps = [int(heads[i+1] - heads[i]) for i in range(len(heads)-1)]
    tail_gaps = [int(tails[i+1] - tails[i]) for i in range(len(tails)-1)]
    head_variety = rao_quadratic_entropy_log(head_gaps)
    tail_variety = rao_quadratic_entropy_log(tail_gaps)
```

```python
    return head_variety + 0.13*tail_variety

# Here acc is acc_v2
def variety_multiplier(acc, v):
    floor = 0.94
    cap = 1.06
    L = cap - floor
    v0 = 3.25
    k = 3

    sigmoid_variety = floor + L / (1 + np.exp(-k * (v - v0)))
    return 1 + 0.1 * (sigmoid_variety - 1) * (5 + np.maximum(0, acc - 95))
```

[2]:

```python
# Test the multiplier function
print("f(96, 2.5):", variety_multiplier(99, 2.5))
print("f(98, 3):",   variety_multiplier(98, 3))
print("f(97, 3.5):", variety_multiplier(97, 3.5))
print("f(97, 4):", variety_multiplier(97, 4))
```

```
f(96, 2.5): 0.9562977422091037
f(98, 3): 0.9827988448791622
f(97, 3.5): 1.0150510107307331
f(97, 4): 1.0339906449484748
```

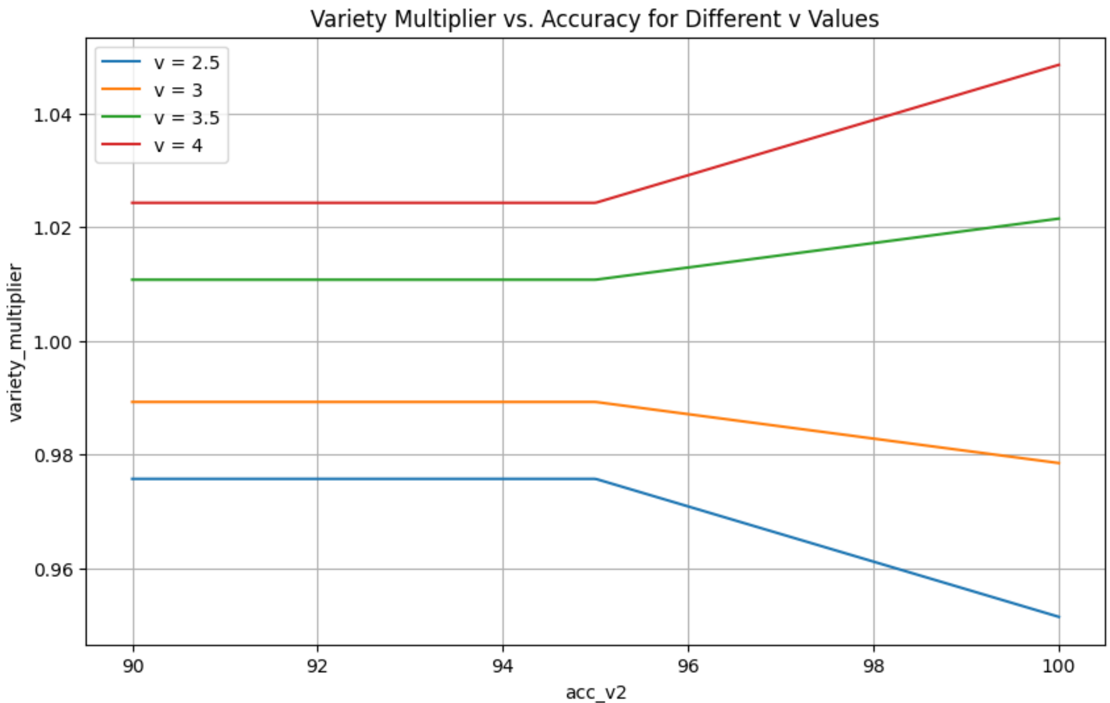
[3]:

```python
acc_values = np.linspace(90, 100, 400)

# Define the list of v values for which we want to graph the function.
v_values = [2.5, 3, 3.5, 4]

plt.figure(figsize=(10, 6))
for v in v_values:
    y = variety_multiplier(acc_values, v)
    plt.plot(acc_values, y, label=f'v = {v}')

plt.xlabel('acc_v2')
plt.ylabel('variety_multiplier')
plt.title('Variety Multiplier vs. Accuracy for Different v Values')
plt.legend()
plt.grid(True)
plt.show()
```


Variety Multiplier vs. Accuracy for Different v Values
v = 2.5
v = 3
v = 3.5
v = 4
1.04
1.02
1.00
0.98
0.96
variety_multiplier
90
92
94
96
98
100
acc_v2

[ ]: