对于视频点播(Youtube)和游戏(CSGO)的流量分类
20337021 陈浚铭

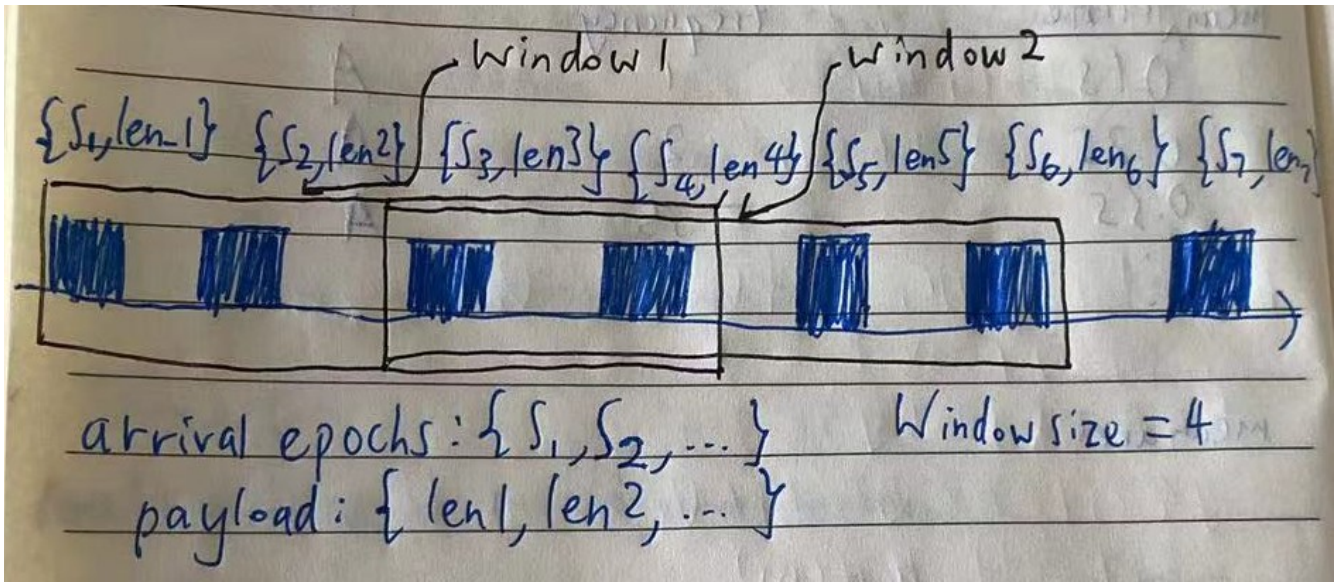
实验所采用的方法和原理：

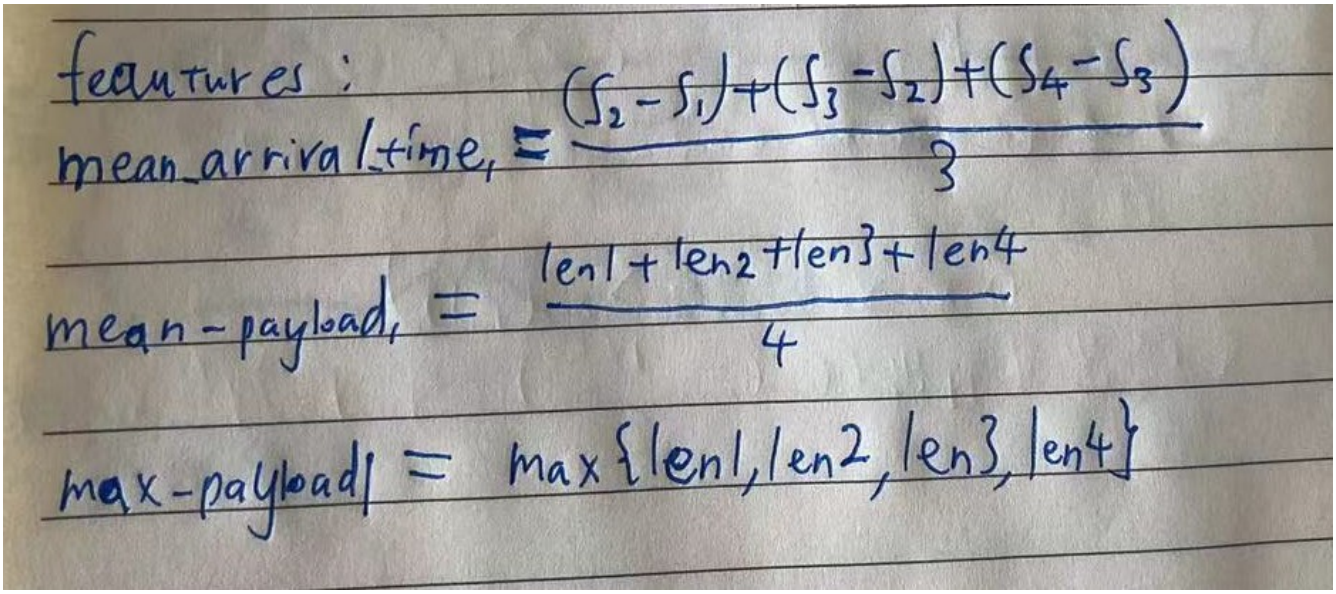
我这个实验要分类的两个类型为视频点播（Youtube)和游戏（game), 我所采用的流量分类方法是基于统计特征的方法[1]，当中分别做了两次实验，一个使用平均时间间隔和平均字节数，另一个使用平均时间间隔和最大字节数，当中发现使用平均时间隔和最大字节数有最好的效果。使用的流是向下流（服务器到客服端）。使用的机器学习模型是 support vector machine。

我认为统计特征的方法有效，是因为视频点播和游戏的时间间隔和字节数这两个特征有显著的 difference。因为，对于 Youtube, 他的向下流的时间间隔的特性是，为了保证用户体验（quality of experience),在播放视频时，同时会不断地缓存未来要播放的视频内容，因此有独有的时间间隔规律（pattern)；而对于字节数的特性，向下流的字节数是比较平均的（1250 byte)；而对于游戏，因为我获取的游戏是 CSGO 这种 multiplayer first player shooter 游戏，对于时间间隔的特性，因此为了它的向下流（以及向下流）的时间间隔是非常短的，以高效地响应目前游戏的状态，而论文[2]提到，CSGO 具有 bursty traffic 的特性，当中 30% 的数据包在一个流量爆发中发送；而对于字节数的特性，我发现字节数的数量并没有规定的大小(数据包大小为 80-319 字节），而在论文[2]提到，CSGO 服务器会根据发送数据需求来增加数据包大小发送给客服端。

因此，因为对于字节数和时间间隔这两个特征，youtube 和 csgo 显著的区别，所以能够使用这两个特征来进行分类。

之后，我们需要讨论**获取特征**的方法。





因此，我们对于**每个流**，分为长度为 4 的窗口，计算平均时间间隔和平均字节数/最大字节数。

机器学习模型： support vector machine

首先，支持向量是最靠近超平面的数据点，意味着它们最难去分类的数据点。

我们被给定一个训练集，它由 n 个点组成：(x_1,y_1),(x_2,y_2), …, (x_n,y_n)，当中 x_i 是一个二维向量，代表平均时间间隔和平均字节数/最大字节数，而每个 y_i 是 1 或者-1, 代表 x_i 所属于的类型(class)。我们想找出最大分类间隔超平面(maximum marginal hyperplane),这个平面能够区分属于 y_i=1 的一组 x_i 和属于 y_i=0 的一组 x_i。这个超平面使得超平面和最近的点（支持向量）最大化。

每一个超平面可以以一组点 x 表示, 它们符合 $w^Tx - b = 0$, 当中 w 是超平面的法线向量，但是 w 不一定是单位向量。参数 b/ ||w||,判定了原点到超平面沿着 w 向量到原点的距离。

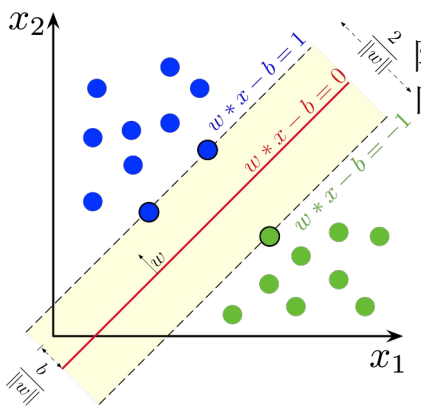


Figure 中显示了两个类型的 SVM 的最大分类间隔超平面和分类间隔。在分类间隔的点就是支持向量。

之后，SVM 分别有 soft-margin 和 hard-margin 的分法，分别对于线性可分和线性不可分的数据来处理。

另外，在我们的流量分类问题，需要分类流量为 Youtube, Game 和 other，因此这种分类问题是多类问题(multiclass classification problem)。为了扩展 SVM 来解决多类问题，我们使用 one vs rest 的 SVM 来解决。

实验代码

过滤流的代码

```
def filter_flow(src_ip, dst_ip, src_port, dst_port, transport_protocol, packet_list, max_num_flows = 15000):
    flow_data_list = []
    flow_counter = 0
    for packet in packet_list:
        if IP in packet and transport_protocol in packet and
        packet[IP].src == src_ip and packet[IP].dst == dst_ip and
        packet[transport_protocol].sport == src_port and packet[transport_protocol].dport == dst_port:
            flow_data_list.append([packet.time, len(packet[transport_protocol].payload)])
            flow_counter = flow_counter + 1
            if flow_counter == max_num_flows:
                break
    return flow_data_list
```

输入特征的函数（窗口大小为 16）

```
# features:
# x1: mean interarrival time
# x2: mean payload / max payload
def add_features(feature_list, label_list,flow_data_list, class_name):
    window_features = []
    num_flows = len(flow_data_list)
    num_features_added = 0
    for i in range(num_flows):
        window_features.append(flow_data_list[i])
        window_size = len(window_features)
        if(window_size == 16 or (i == num_flows - 1)):
            # compute features
            mean_interarrival_time = 0
            mean_payload = 0
            arrival_time_list = []
            max_payload = 0
            for features in window_features:
                arrival_time_list.append(features[0])
                mean_payload = mean_payload + features[1]
                if(max_payload < features[1]):
                    max_payload = features[1]

            mean_payload = mean_payload / window_size
            for i in range(window_size-1):
                mean_interarrival_time = mean_interarrival_time + (arrival_time_list[i+1] - arrival_time_list[i])
            mean_interarrival_time = mean_interarrival_time / (window_size-1)
            window_features = window_features[8:]
            feature_list.append([mean_interarrival_time, max_payload])
            label_list.append(class_name)
            num_features_added = num_features_added + 1
    return num_features_added
```

在 main 函数读取流：

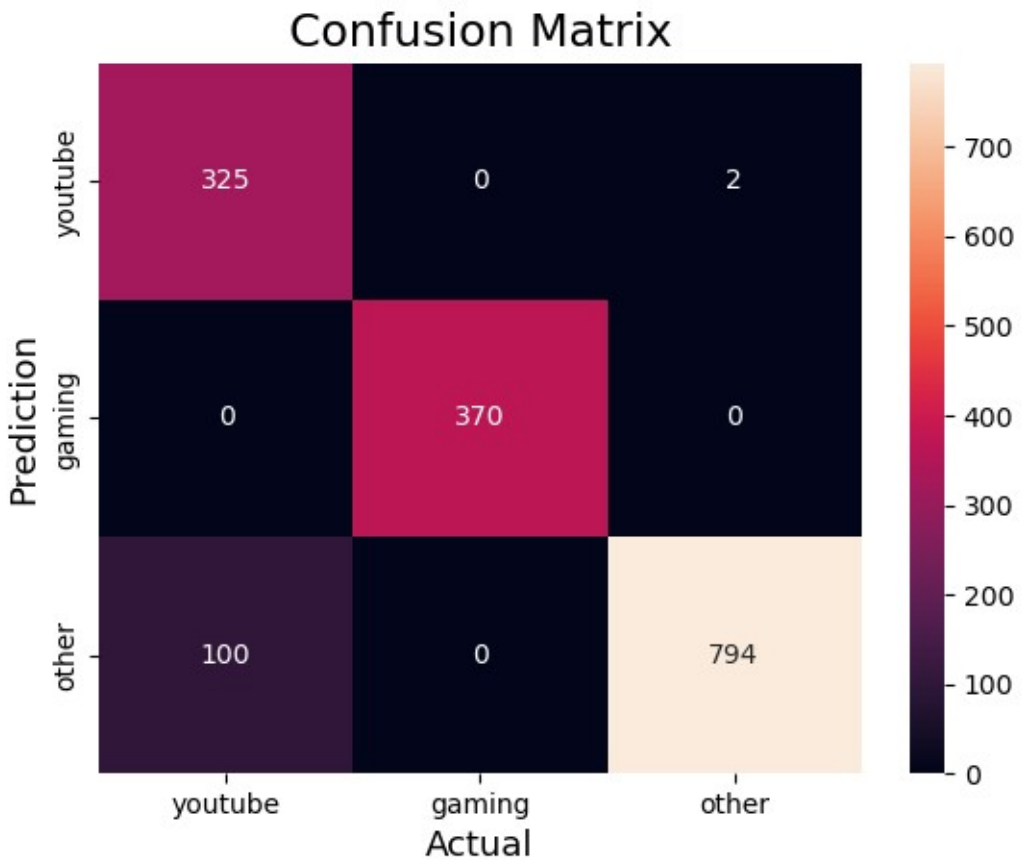
```
    youtube_packets = rdpcap('youtube_flows.pcapng')
    flow_data_list_1 = filter_flow('58.176.217.144', '192.168.128.72', 443, 64511, UDP, youtube_packets)
    num_features_1 = add_features(feature_list, label_list, flow_data_list_1, "youtube")
    print(num_features_1)

    game_packets = rdpcap('game_flows.pcapng')
    flow_data_list_2 = filter_flow('117.162.37.59','172.26.32.132', 16285, 49669, UDP, game_packets)
    num_features_2 = add_features(feature_list, label_list, flow_data_list_2, "gaming")
    print(num_features_2)
```

另外还有其他读取其他类型的流: 高校网站（school.pcapng),
商业网站(ecommerce.pcapng)，线上直播(streaming.pcapng)

实验结果：

我没有进行复杂度分析，因为数据集非常小，运行时间和内存消耗**微不足道**

对于使用**平均时间间隔**和**平均字节数**：
混淆矩阵

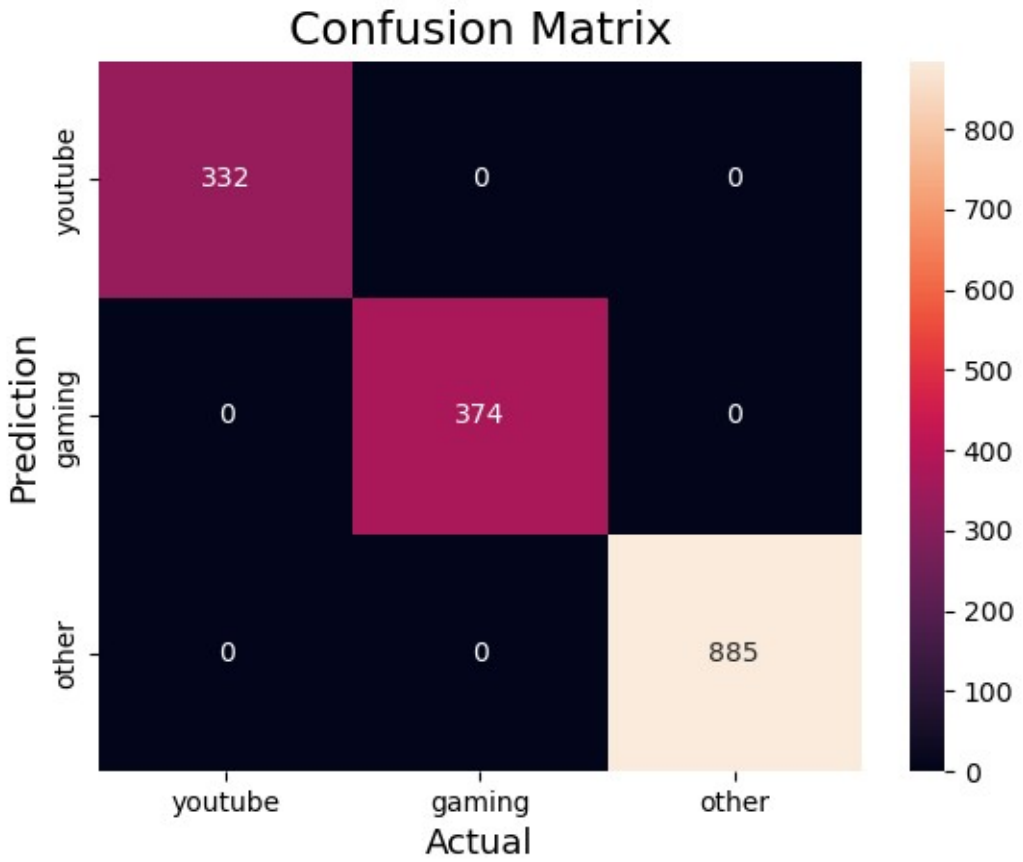


准确率，误判率，F1-Score:

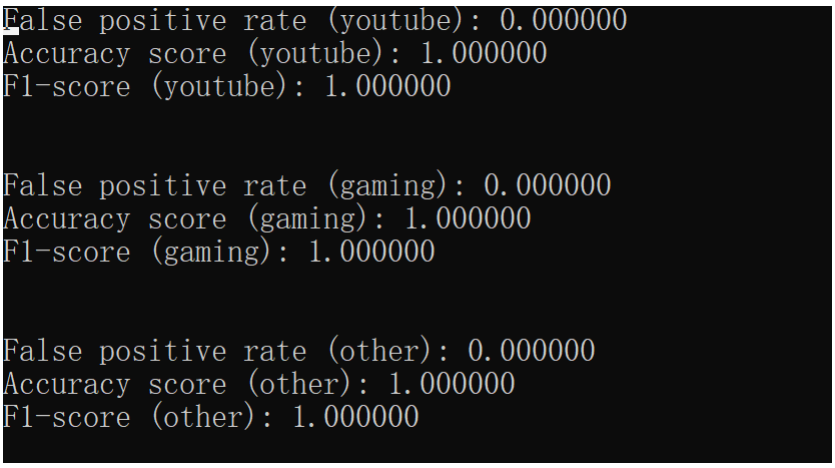
```
False positive rate (youtube): 0.079114
Accuracy score (youtube): 0.935889
F1-score (youtube): 0.864362


False positive rate (gaming): 0.000000
Accuracy score (gaming): 1.000000
F1-score (gaming): 1.000000


False positive rate (other): 0.002869
Accuracy score (other): 0.935889
F1-score (other): 0.939645
```

对于使用**平均时间间隔**和**最大平均字节数**

混淆矩阵



准确率，误判率，F1-Score:



因此，我们发现，使用**平均时间间隔**和**最大平均字节数**作为特征能够**百分之百**地分类数据，因此比使用**平均时间间隔**和平均**平均字节数**作为特征更有效。

参考文献

[1] A. Azab et al., Network traffic classification: Techniques, datasets, and challenges, Digital Communications and Networks, https://doi.org/10.1016/j.dcan.2022.09.009

[2]M. Claypool, D. LaPoint and J. Winslow, "Network analysis of Counter-strike and Starcraft," *Conference Proceedings of the 2003 IEEE International Performance, Computing, and Communications Conference, 2003.*, Phoenix, AZ, USA, 2003, pp. 261-268, doi: 10.1109/PCCC.2003.1203707.