2024 鐵人賽 – 我數學就爛要怎麼來學 DNN 模型安全
Day 29 – Clean Label Attack

大綱

- Clean Label 攻擊
  - 前情提要
  - 論文演算法
  - 程式實作

- 結論

前情提要

https://github.com/ashafahi/inceptionv3-transferLearn-poison

- 看似做完了論文的最佳化式子，其實並沒有

Let $f(\mathbf{x})$ denote the function that propagates an input $\mathbf{x}$ through the network to the penultimate layer (before the softmax layer). We call the activations of this layer the *feature space* representation of the input since it encodes high-level semantic features. Due to the high complexity and nonlinearity of f, it is possible to find an example $\mathbf{x}$ that "collides" with the target in feature space, while simultaneously being close to the base instance $\mathbf{b}$ in input space by computing

$$\mathbf{p} = \underset{\mathbf{x}}{\operatorname{argmin}} \ \|f(\mathbf{x}) - f(\mathbf{t})\|_2^2 + \beta \|\mathbf{x} - \mathbf{b}\|_2^2 \tag{1}$$

The right-most term of Eq. 1 causes the poison instance $\mathbf{p}$ to appear like a base class instance to a human labeler (β parameterizes the degree to which this is so) and hence be labeled as such.

論文演算法

https://arxiv.org/abs/1411.3406

- 論文的後半部有一個神奇的演算法虛擬碼

distance from the base instance in input space. The coefficient β is tuned to make the poison instance look realistic in input space, enough to fool an unsuspecting human observer into thinking the attack vector image has not been tampered with.

Algorithm 1 Poisoning Example Generation

Input: target instance t, base instance b, learning rate λ
Initialize x: $x_0 \leftarrow b$
Define: $L_p(x) = \|f(\mathbf{x}) - f(\mathbf{t})\|^2$
for $i = 1$ **to** $maxIters$ **do**
  Forward step: $\widehat{x_i} = x_{i-1} - \lambda \nabla_x L_p(x_{i-1})$
  Backward step: $x_i = (\widehat{x_i} + \lambda\beta b)/(1 + \beta\lambda)$
end for

論文演算法

$$\mathbf{p} = \underset{\mathbf{x}}{\operatorname{argmin}} \; \|f(\mathbf{x}) - f(\mathbf{t})\|_2^2 + \beta \|\mathbf{x} - \mathbf{b}\|_2^2$$

- 分析一下式子，從原本兩個式子的最佳化變成一個

distance from the base instance in input space. The coefficient β is tuned to make the poison instance look realistic in input space, enough to fool an unsuspecting human observer into thinking the attack vector image has not been tampered with.

Algorithm 1 Poisoning Example Generation

Input: target instance t, base instance b, learning rate λ
Initialize x: $x_0 \leftarrow b$
Define: $L_p(x) = \|f(\mathbf{x}) - f(\mathbf{t})\|^2$
for $i = 1$ **to** $maxIters$ **do**
  Forward step: $\widehat{x}_i = x_{i-1} - \lambda \nabla_x L_p(x_{i-1})$
  Backward step: $x_i = (\widehat{x}_i + \lambda\beta b)/(1 + \beta\lambda)$
end for

論文演算法

- 這個神奇的算法我看了很久，發現 β 越大與原圖的差異占比越多，所以整體 loss 會傾向跟原圖類似

$$\text{Forward step: } \widehat{x_i} = x_{i-1} - \lambda \nabla_x L_p(x_{i-1})$$
$$\text{Backward step: } x_i = (\widehat{x_i} + \lambda\beta b)/(1+\beta\lambda)$$

簡化變數觀察一下

$\beta = 1, \lambda = 1 \rightarrow X_i = (\frac{1}{2}\hat{X}_i + \frac{1}{2}b)$

$\beta = 3, \lambda = 1 \rightarrow X_i = (\frac{1}{4}\hat{X}_i + \frac{3}{4}b)$

程式實作

https://github.com/ashafahi/inceptionv3-transferLearn-poison

- 其實反而比較簡單，最佳化式子剩一個，其餘就是給值
- 給值要注意用 assign 而非用 = ，否則會悲劇

```
for i in range(1000) :

    with tf.GradientTape(persistent=True) as tape:
        #loss1 = tf.keras.losses.MeanSquaredError(reduction='sum')(x,t_base_image)
        loss2 = tf.keras.losses.MeanSquaredError(reduction='sum')(new_model(x),new_model(t_target_image))
        #loss = loss1 + c*loss2
        loss = loss2

    gradients = tape.gradient(loss, x)
    # 這邊依然要注意負值的問題
    x.assign(tf.clip_by_value(x,0,1))

    #print(gradients)

    tf.optimizers.Adam(learning_rate=learning_rate).apply_gradients(zip([gradients],[x]))
    x.assign((x+beta*learning_rate*t_base_image)/(1+beta*learning_rate))
```

結論

- 最佳化數值的式子簡化後還是會有負值問題，看來真的要實做的時候自行解決
- 但是在式子的簡化上原本有兩個要求最佳解的式子轉變成只有一個，只能說數學真的很厲害