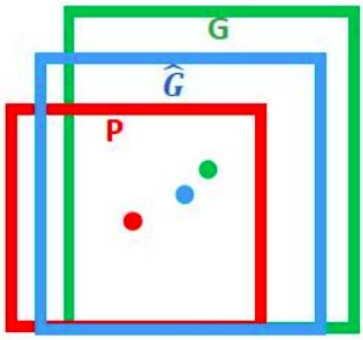
一、问题的数学表达



对于窗口，一般使用(x,y,w,h)来表示，分别表示窗口的中心点坐标和宽高。

对于上图，红色的框P表示原始的$proposal$候选目标框，蓝色的框$\hat{G}$代表边界框回归算法预测的目标框，绿色的框G代表目标的$Ground\quad Truth$真实目标框。
红色圆圈代表候选目标框的中心点，绿色圆圈代表真实目标框的中心点，蓝色圆圈代表边界框回归算法预测目标框的中心点。
我们的目标是寻找一种映射关系，使得原始窗口P经过映射后得到一个跟真实窗口G更接近的回归窗口$\hat{G}$。

首先先对边界框回归的输入数据集进行说明。

输入到边界框回归的数据集为$\{(P^i,G^i)\}_{i=1,2,\cdots,N}$，其中$P^i=(P_x^i,P_y^i,P_w^i,P_h^i)$，$G^i=(G_x^i,G_y^i,G_w^i,G_h^i)$。$P^i$代表第$i$个带预测的候选目标检测框即 region proposal。G^i是第i个真实目标检测框即 ground-truth。在 R-CNN 和 Fast R-CNN 中，P^i是利用选择性搜索算法进行获取。

在P^i中，P_x^i代表候选目标框的中心点在原始图像中的x坐标，P_y^i代表候选目标框的中心点在原始图像中的y坐标，P_w^i代表候选目标框的长度，P_h^i代表候选目标框的宽度。G^i的四维特征的含义与P^i是一样的。

那么边界框回归要做的就是利用某种映射关系，使得候选目标框（region proposal）的映射目标框无限接近于真实目标框（ground-truth）。将上述原理利用数学符号表示如下：

在给定一组候选目标框$P=(P_x,P_y,P_w,P_h)$，寻找到一个映射f，使得：

$$f(P_x,P_y,P_w,P_h)=(\hat{G}_x,\hat{G}_y,\hat{G}_w,\hat{G}_h)\approx(G_x,G_y,G_w,G_h)$$

二、Bounding-box regression

那么经过何种变换才能从上图的窗口 P 变为窗口 $\widehat{G}$ 呢?

比较简单的思路就是:

(1) 先做平移 $(\Delta x, \Delta y)$， $\Delta x = P_w d_x(P)$， $\Delta y = P_h d_y(P)$

这实际上就是 R-CNN 论文里面的:

$$\widehat{G}_x = P_w d_x(P) + P_x$$

$$\widehat{G}_y = P_h d_y(P) + P_y$$

(2) 然后再做尺度缩放 (S_w, S_h)， $S_w = P_w d_w(P)$， $S_h = P_h d_h(P)$

这实际上就是 R-CNN 论文里面的:

$$\widehat{G}_w = P_w \exp(d_w(P))$$

$$\widehat{G}_h = P_h \exp(d_h(P))$$

观察上述公式我们发现，我们需要学习的是 $d_x(P)$， $d_y(P)$， $d_w(P)$， $d_h(P)$ 这四个变换。

下一步就是设计算法得到这四个映射。

当输入的 proposal 与 Ground Truth 相差较小时，可以认为这种变换是一种线性变换，那么我们就可以用线性回归来建模对窗口进行微调。

注意:

只有当 proposal 和 Ground Truth 比较接近时，我们才能将其作为训练样本训练我们的线性回归模型，否则导致训练的回归模型不 work（当 proposal 跟 GT 离得较远时，就是复杂的非线性问题了，此时用线性回归建模显然不合理）。

线性回归就是给定输入的特征向量 X，学习一组参数 W，使得经过线性回归后的值和真实值 Y 非常接近，即 $Y \approx WX$。那么边界框回归中我们的输入以及输出分别是什么呢?

输入：Region Proposal -> $P = (P_x, P_y, P_w, P_h)$。输入就是这四个数值吗？其实真正的输入就是这个窗口对应的 CNN 特征，也就是 R-CNN 中的 $pool_5$ $feature$。

训练阶段输入还包括 Ground Truth，也就是下面提到的 $t_* = (t_x, t_y, t_w, t_h)$。

输出：需要进行的平移变换和尺度缩放 $d_x(P)$， $d_y(P)$， $d_w(P)$， $d_h(P)$，或者说是 Δx，Δy， S_w， S_h。

有了这四个变换，我们就可以直接得到 Ground Truth，这里还有个问题，根据 R-CNN 论文里面的公式我们可以知道， P 经过 $d_x(P)$， $d_y(P)$， $d_w(P)$， $d_h(P)$ 得到的并不是真实值 G，而是预测值 $\widehat{G}$。的确，这四个值应该是经过 Ground Truth 和 Proposal 计算得到的真正

需要的平移量(t_x, t_y)和尺度缩放(t_w, t_h)。

这也是 R-CNN 中提到的公式：

$$t_x = (G_x - P_x) / P_w$$

$$t_y = (G_y - P_y) / P_h$$

$$t_w = \log(G_w / P_w)$$

$$t_h = \log(G_h / P_h)$$

我们用$t_* = (t_x, t_y, t_w, t_h)$对$d_x(P)$，$d_y(P)$，$d_w(P)$，$d_h(P)$进行监督。

那么目标函数可以表示为：$d_*(P) = w_*^T \Phi_5(P)$

$\Phi_5(P)$是输入 Proposal 的特征向量，w_*是要学习的参数（*表示x, y, w, h，也就是每一个变换对应一个目标函数），$d_*(P)$是得到的预测值。我们要让预测值跟真实值$t_* = (t_x, t_y, t_w, t_h)$差距最小，得到损失函数为：

$$Loss = \sum_i^N (t_*^i - \widehat{w_*}^T \phi_5(P^i))^2$$

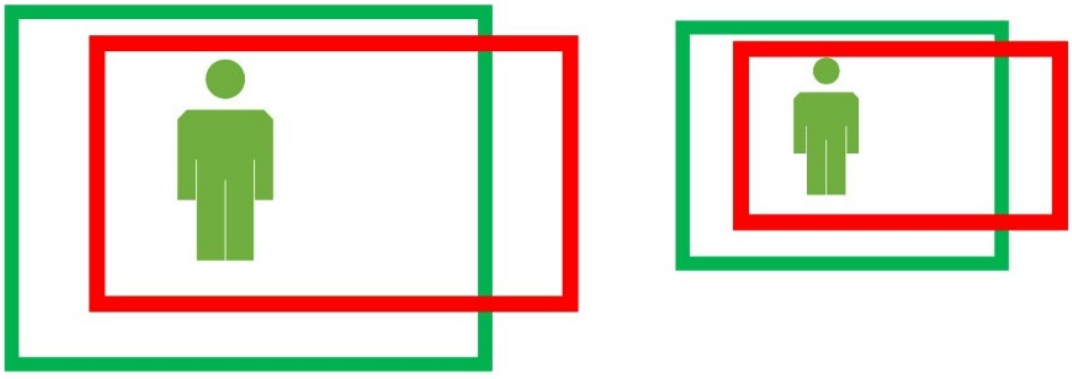
函数优化目标为：

$$w_* = \arg\min_{\widehat{w_*}} \sum_i^N (t_*^i - \widehat{w_*}^T \phi_5(P^i))^2 + \lambda \| \widehat{w_*} \|^2$$

利用梯度下降法或者最小二乘法就可以得到w_*。

三、为什么使用相对坐标差

在上式中，为什么要把真实框的中心坐标与候选框的中心坐标的差值分别除以宽高呢？首先我们假设两张尺寸不同，但内容相同的图像，图像如下图所示：



那么我们假设经过 CNN 提取到的特征分别是ϕ_1和ϕ_2。同时，我们假设x_i为第i个真实目标框的x坐标，p_i为第i个候选目标框的x坐标，边界框回归的映射关系为f。那么我们可以得出：

$$\begin{cases} f(\phi_1) = x_1 - p_1 \\ f(\phi_2) = x_2 - p_2 \end{cases}$$

由于 CNN 具有尺度不变性，因此$\phi_1 = \phi_2$。那么理论上$x_1 - p_1 = x_2 - p_2$。但是观察上图就可明显得出$x_1 - p_1 \neq x_2 - p_2$，显然由于尺寸的变化，候选目标框和真实目标框坐标之间的偏移量也随着尺寸而成比例缩放，即这个比例值是恒定不变的。
因此，我们必须对x坐标的偏移量除以候选目标框的宽，y坐标的偏移量除以候选目标框的高。只有这样才能得到候选目标框与真实目标框之间坐标偏移量的相对值。同时使用相对偏移量的好处可以自由选择输入图像的尺寸，使得模型灵活多变，也就是说，对坐标偏移量除以宽高就是在做尺度归一化，即尺寸较大的目标框的坐标偏移量较大，尺寸较小的目标框的坐标偏移量较小。

四、为什么宽高要取对数?

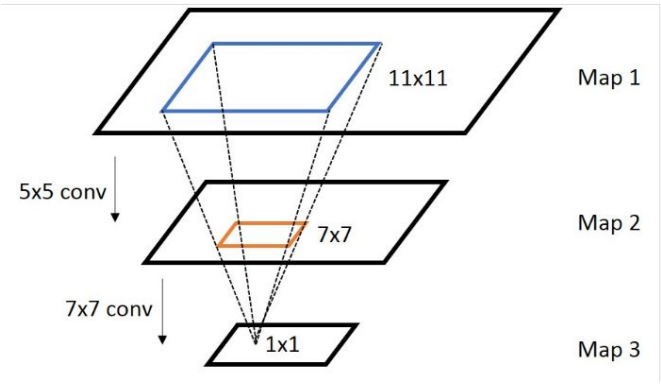
我们来看看预测框和真实框的真正的差：

$$\begin{cases} t_w = \log(\frac{G_w}{P_w}) = \log(\frac{G_w + P_w - P_w}{P_w}) = \log(1 + \frac{G_w - P_w}{P_w}) \\ t_h = \log(\frac{G_h}{P_h}) = \log(\frac{G_h + P_h - P_h}{P_h}) = \log(1 + \frac{G_h - P_h}{P_h}) \end{cases}$$

我们知道在高等数学中，$\lim_{x \to 0} \frac{\log(1+x)}{x} = 1$，二者是等价无穷小。所以当$G_w$和$P_w$非常接近的时候，上面这个函数是个近似的线性函数。所以是可以用线性回归来近似。个人理解，只保证 region proposal 和 Ground Truth 的宽高相差不多就能满足回归条件。

五、感受野（卷积层的输入）上面的坐标映射

output field size = (input field size - kernel size + 2*padding) / stride + 1
卷积层的输入（也即前一层的感受野） = ?
input field size = （output field size - 1）* stride - 2*padding + kernel size
通常，我们需要知道网络里面任意两个 feature map 之间的坐标映射关系（一般是中心点之间的映射），如下图，我们想得到 map 3 上的点 p3 映射回 map 2 所在的位置 p2（橙色框的中心点）

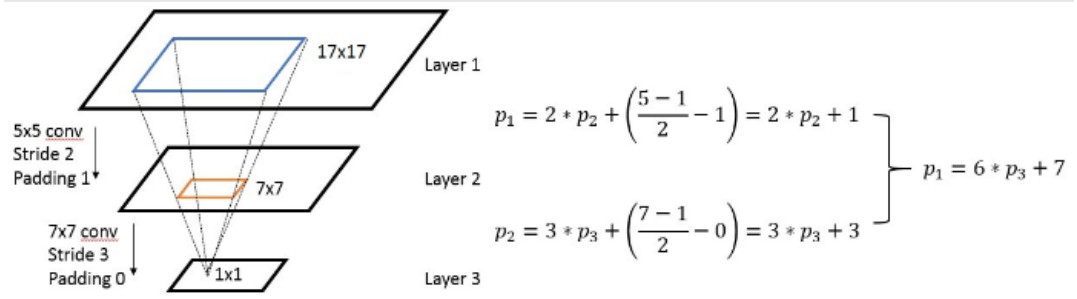


计算公式：

$Convolution / Pooling \quad layer$ **：** $p_i = s_i \cdot p_{i+1} + ((k_i - 1)/2 - padding)$

$Neuronlayer(\mathrm{Re}LU / Sigmoid / .....)$ **：** $p_I = p_{i+1}$

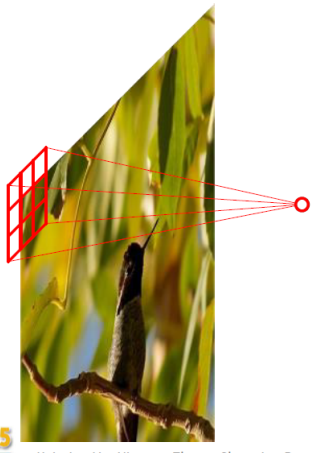
上面是计算任意一个 layer 输入输出的坐标映射关系，如果是计算任意 feature map 之间的关系，只需要用简单的组合就可以得到，下面是一个简单的例子。



最后说一下，那张 PPT 里面的公式：

Microsoft Research

Receptive Field



How to compute the center of the receptive field

- A simple solution
  - For each layer, pad $\lfloor F/2 \rfloor$ pixels for a filter size F (e.g., pad 1 pixel for a filter size of 3)
  - On each feature map, the response at (0, 0) has a receptive field centered at (0, 0) on the image
  - On each feature map, the response at (x, y) has a receptive field centered at (Sx, Sy) on the image (stride S)
- A general solution

$$i_0 = g_L(i_L) = \alpha_L(i_L - 1) + \beta_L,$$
$$\alpha_L = \prod_{p=1}^{L} s_p,$$
$$\beta_L = 1 + \sum_{p=1}^{L} \left(\prod_{q=1}^{p-1} s_q\right)\left(\frac{F_p - 1}{2} - P_p\right)$$

See [Karel Lenc & Andrea Vedaldi] "R-CNN minus R". BMVC 2015.

ICCV15 International Conference on Computer Vision

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition". ECCV 2014.

对于上面的 A simple solution：其实也是何凯明在 SPP-net 中采用的方法。其实就是巧妙地化简一下公式： $p_i = s_i \cdot p_{i+1} + ((k_i - 1)/2 - padding)$ 令每一层的 padding 都为 $padding = \lfloor k_i / 2 \rfloor \Rightarrow p_i = s_i \cdot p_{i+1} + ((k_i - 1)/2 - \lfloor k_i / 2 \rfloor)$

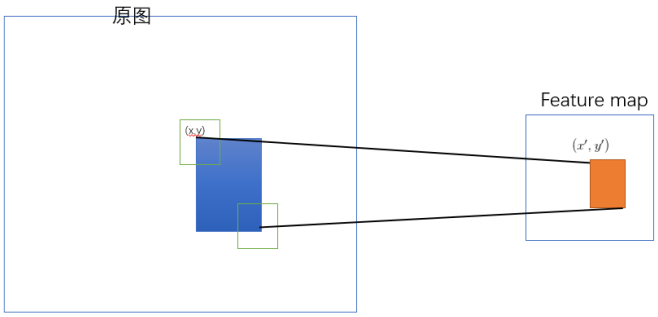
当 k_i 为奇数时，$((k_i - 1)/2 - \lfloor k_i / 2 \rfloor) = 0$ 所以 $p_i = s_i \cdot p_{i+1}$

当 k_i 为偶数时，$((k_i - 1)/2 - \lfloor k_i / 2 \rfloor) = -0.5$，所以 $p_i = s_i \cdot p_{i+1} - 0.5$

因为 p_i 是图像中的坐标点，不能取小数，所以基本上可以认为 $p_i = s_i \cdot p_{i+1}$。公式得到了化简：感受野中心点的坐标 p_i 只跟前一层 p_{i+1} 有关。

SPP-Net 的 ROI 映射做法详解

SPP-Net 是把原始 ROI 的左上角和右下角映射到 feature map 上的两个对应点。有了 feature map 上的两队角点就确定了对应的 feature map 区域。



如何映射?

左上角的点（x,y）映射到 feature map上的(x', y')： 使得 (x', y') 在原始图上感受野（上图绿色框）的中心点 与（x,y）尽可能接近。

对应点之间的映射公式是啥?

- 就是前面每层都填充padding/2 得到的简化公式： $p_i = s_i \cdot p_{i+1}$
- 需要把上面公式进行级联得到 $p_0 = S \cdot p_{i+1}$ 其中 $(S = \prod_0^i s_i)$
- 对于feature map 上的 (x', y') 它在原始图的对应点为 $(x, y) = (Sx', Sy')$
- 论文中的最后做法：把原始图片中的ROI映射为 feature map中的映射区域（上图橙色区域）其中 左上角取：$x' = \lfloor x/S \rfloor + 1,\ y' = \lfloor y/S \rfloor + 1,$；右下角的点取： 界取y'的x值：$x' = \lceil x/S \rceil - 1,\ y' = \lceil y/S \rceil - 1$。 下图可见 $\lfloor x/S \rfloor + 1$, $\lceil x/S \rceil - 1$ 的作用效果分别是增加和减少。也就是 左上角要向右下偏移，右下角要想要向左上偏移。个人理解采取这样的策略是因为论文中的映射方法（左上右下映射）会导致feature map上的区域反映射回原始ROI时有多余的区域（下图左边红色框是比蓝色区域大的）