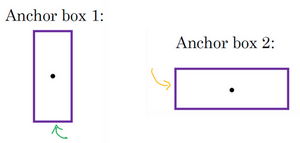
### 3.8 Anchor Boxes

到目前为止，对象检测中存在的一个问题是每个格子只能检测出一个对象，如果你想让一个格子检测出多个对象，你可以这么做，就是使用**anchor box**这个概念，我们从一个例子开始讲吧。

图片包含 天空

描述已自动生成

假设你有这样一张图片，对于这个例子，我们继续使用3×3网格，注意行人的中点和汽车的中点几乎在同一个地方，两者都落入到同一个格子中。所以对于那个格子，如果 输出这个向量，你可以检测这三个类别，行人、汽车和摩托车，它将无法输出检测结果，所以我必须从两个检测结果中选一个。



而**anchor box**的思路是，这样子，预先定义两个不同形状的**anchor box**，或者**anchor box**形状，你要做的是把预测结果和这两个**anchor box**关联起来。一般来说，你可能会用更多的**anchor box**，可能要5个甚至更多，但对于这个视频，我们就用两个**anchor box**，这样介绍起来简单一些。

图片包含 文字, 白板

描述已自动生成

你要做的是定义类别标签，用的向量不再是上面这个：

而是重复两次：

前面的（绿色方框标记的参数）是和**anchor box 1**关联的8个参数，后面的8个参数（橙色方框标记的元素）是和**anchor box 2**相关联。因为行人的形状更类似于**anchor box 1**的形状，而不是**anchor box 2**的形状，所以你可以用这8个数值（前8个参数），这么编码，是的，代表有个行人，用和来编码包住行人的边界框，然后用()来说明这个对象是个行人。

然后是车子，因为车子的边界框比起**anchor box 1**更像**anchor box 2**的形状，你就可以这么编码，这里第二个对象是汽车，然后有这样的边界框等等，这里所有参数都和检测汽车相关()。

图片包含 人员, 文字

描述已自动生成

总结一下，用**anchor box**之前，你做的是这个，对于训练集图像中的每个对象，都根据那个对象中点位置分配到对应的格子中，所以输出就是3×3×8，因为是3×3网格，对于每个网格位置，我们有输出向量，包含，然后边界框参数和，然后。

现在用到**anchor box**这个概念，是这么做的。现在每个对象都和之前一样分配到同一个格子中，分配到对象中点所在的格子中，以及分配到和对象形状交并比最高的**anchor box**中。所以这里有两个**anchor box**，你就取这个对象，如果你的对象形状是这样的（编号1，红色框），你就看看这两个**anchor box**，**anchor box 1**形状是这样（编号2，紫色框），**anchor box 2**形状是这样（编号3，紫色框），然后你观察哪一个**anchor box**和实际边界框（编号1，红色框）的交并比更高，不管选的是哪一个，这个对象不只分配到一个格子，而是分配到一对，即（**grid cell，anchor box**）对，这就是对象在目标标签中的编码方式。所以现在输出 就是3×3×16，上一张幻灯片中你们看到 现在是16维的，或者你也可以看成是3×3×2×8，因为现在这里有2个**anchor box**，而 是8维的。 维度是8，因为我们有3个对象类别，如果你有更多对象，那么 的维度会更高。

图片包含 文字

描述已自动生成

所以我们来看一个具体的例子，对于这个格子（编号2），我们定义一下，：

。

所以行人更类似于**anchor box 1**的形状，所以对于行人来说，我们将她分配到向量的上半部分。是的，这里存在一个对象，即，有一个边界框包住行人，如果行人是类别1，那么 （编号1所示的橙色参数）。车子的形状更像**anchor box 2**，所以这个向量剩下的部分是 ，然后和车相关的边界框，然后（编号1所示的绿色参数）。所以这就是对应中下格子的标签 ，这个箭头指向的格子（编号2所示）。

现在其中一个格子有车，没有行人，如果它里面只有一辆车，那么假设车子的边界框形状是这样，更像**anchor** **box 2**，如果这里只有一辆车，行人走开了，那么**anchor box 2**分量还是一样的，要记住这是向量对应**anchor box 2**的分量和**anchor box 1**对应的向量分量，你要填的就是，里面没有任何对象，所以 ，然后剩下的就是**don’t care-s**(即？)（编号3所示）。

现在还有一些额外的细节，如果你有两个**anchor box**，但在同一个格子中有三个对象，这种情况算法处理不好，你希望这种情况不会发生，但如果真的发生了，这个算法并没有很好的处理办法，对于这种情况，我们就引入一些打破僵局的默认手段。还有这种情况，两个对象都分配到一个格子中，而且它们的**anchor box**形状也一样，这是算法处理不好的另一种情况，你需要引入一些打破僵局的默认手段，专门处理这种情况，希望你的数据集里不会出现这种情况，其实出现的情况不多，所以对性能的影响应该不会很大。

这就是**anchor box**的概念，我们建立**anchor box**这个概念，是为了处理两个对象出现在同一个格子的情况，实践中这种情况很少发生，特别是如果你用的是19×19网格而不是3×3的网格，两个对象中点处于361个格子中同一个格子的概率很低，确实会出现，但出现频率不高。也许设立**anchor box**的好处在于**anchor box**能让你的学习算法能够更有征对性，特别是如果你的数据集有一些很高很瘦的对象，比如说行人，还有像汽车这样很宽的对象，这样你的算法就能更有针对性的处理，这样有一些输出单元可以针对检测很宽很胖的对象，比如说车子，然后输出一些单元，可以针对检测很高很瘦的对象，比如说行人。

最后，你应该怎么选择**anchor box**呢？人们一般手工指定**anchor box**形状，你可以选择5到10个**anchor box**形状，覆盖到多种不同的形状，可以涵盖你想要检测的对象的各种形状。还有一个更高级的版本，我就简单说一句，你们如果接触过一些机器学习，可能知道后期**YOLO**论文中有更好的做法，就是所谓的**k-平均算法**，可以将两类对象形状聚类，如果我们用它来选择一组**anchor box**，选择最具有代表性的一组**anchor box**，可以代表你试图检测的十几个对象类别，但这其实是自动选择**anchor box**的高级方法。如果你就人工选择一些形状，合理的考虑到所有对象的形状，你预计会检测的很高很瘦或者很宽很胖的对象，这应该也不难做。

所以这就是**anchor box**，在下一个视频中，我们把学到的所有东西一起融入到**YOLO**算法中。