Machine or Deep Learning

冯迁

2017 年 11 月 9 日

目录

前言

本文档是我的一个学习过程，学习时间开始于 2016 年末，打算将其写出来于 2017 年中。本着不是科班出生，很多方面都不太扎实，包括程序设计，对算法的性能分析，以及囿于匮乏的经验对市场的状况认识还不太合格。将其写出来一来梳理自己，二来也可以利用自己的数学优势以及先行几步减少部分初学者学习上的困惑，最后也能知道自己的不足之处以及理解有误的地方。很多内容都将会根据网络资源进行说明，补充，综合。

首先以词条人工智能开始维基百科之旅: 人工智能。

AI 的核心问题包括推理, 知识, 规划, 学习, 交流, 感知, 移动和操作物体等。目前比较流行的方法包括统计方法，计算智能和传统意义的 AI。大量应用的人工智能包括搜索和数学优化, 逻辑推演。

接着自由选择进入机器学习页面：机器学习是人工智能的一个分支。在 30 多年的发展中，已成为一门多领域的交叉学科，涉及概率论，统计学，逼近论，凸分析，计算复杂性理论等多门学科。其算法主要实现从数据中自动分析获得规律，并利用规律对未知数据进行预测。主要分为四类：监督学习 (回归分析和统计分类)，无监督学习 (聚类)，半监督学习和增强学习 (基于环境行动，以取得最大化的预期利益)。

具体的机器学习算法:

1. 构造间隔理论分布: 聚类分析和模式识别
   (a) 人工神经网络
   (b) 决策树
   (c) 感知器
   (d) 支持向量机
   (e) 集成学习 AdaBoost
   (f) 降维与度量学习
   (g) 聚类
   (h) 贝叶斯分类器
2. 构造条件概率: 回归分析和统计分类
   (a) 高斯过程回归

   (b) 线性判别分析
   (c) 最近邻法
   (d) 径向基函数核
3. 通过再生模型构造概率密度函数:
   (a) 最大期望算法
   (b) 概率图模型: 贝叶斯网和 Markov 随机场
   (c) Generative Topographic Mapping
4. 近似推断技术:
   (a) 马尔科夫链
   (b) 蒙特卡洛方法
   (c) 变分法
5. 最优化: 大多数以上方法，直接或间接使用最优化算法。

以上是中文版的粗略分类，关于更详尽的分类可参考:Outline_of_ml. 假设我们已经浏览了此页面上所有内容，包括其链接内容。这样我们对人工智能整体有了个一个简单的认识: 算法庞杂，理论繁多，近几年活跃度很大。一方面方法可以很简单，比如 KNN,PCA,BP..., 另一方面也可以很复杂, 比如变分法，最优传输等。以现有的目光来看，很多基本的问题还需要学者们去解答。包括一个统一的理论框架，对深度黑箱的解释等。另外如此多的算法，在面对实际问题时，往往局限于模型的理想化，以及问题的类型，需要根据实际选择并改装。这也就促使我们选择自己感兴趣的分支，并掌握所需算法的精髓。似乎一下就变成了生命能承受之重了，... 似轻非轻..., 道路很长。

* 选择：CV
* 工具:TensorFlow,Pytorch,sklearn,Numpy,Opencv...
* 理论: 相似与度量学习, 凸优化, 概率统计, 最优传输
* 问题: 怎样学到最少的东西, 解决更多的问题?

关于工具的问题，纯粹学习工具本身，是一个挺无聊的过程。带着问题或者项目学习，在一定程度上能减少一些莫名的痛苦。学习工具之前要有一定的理论基础，单单学习框架是无意义的，假设不从长远来看的话，那就没问题了。理论学习若能结合具体问题进行比较，在面向工程时，感觉会过渡得更加自然，反过来，也可能会给理论研究注入新鲜东西。

第一章　数据处理

数据处理是机器学习的关键一步，不论是在训练前还是在训练当中，都存在对数据的各种处理。训练前，数据收集，数据质量评判，数据表示，数据特征抽取，数据降维，数据归一化，训练中，数据批归一化，数据重构...。

接下来假设我们已经拥有了比较完整均匀的数据。

数据预处理

Multivariate Statistical Analysis

多变量分析主要用于分析拥有多个变数的资料，探讨资料彼此之间的关联性或是厘清资料的结构，而有别于传统统计方法所着重的参数估计以及假设检定。常见的分析方法有 PCA,CCA,MDS,SEM 等。

PCA

主成分分析:PCA

PCA 分析计算的核心就是矩阵的奇异值分解，奇异值分解属于谱定理的一小部分，数学上谱定理是个很精彩的定理，但这里我们只能介绍 SVD。

假设 M 是一个 $m \times n$ 阶矩阵，其中的元素全部属于域 K，也就是实数域或复数域。如此则存在一个分解使得

$$M = U\Sigma V^*$$

其中 U 是 $m \times m$ 阶酉矩阵； 是 $m \times n$ 阶非负实数对角矩阵；而 V^*，即 V 的共轭转置，是 $n \times n$ 阶酉矩阵。这样的分解就称作 M 的奇异值分解。Σ 对角线上的元素 Σ_i, i 即为 M 的奇异值。

对于 PCA，我们要分解的就是数据的经验协方差阵，因为协方差阵是对称的，在线性代数里，我们知道每个正规矩阵都可以被一组特征向量对角化。即:

$$M = U\Sigma U^*.$$

实际上对于对称矩阵我们还可以做到

$$M = U\Sigma U^{-1}.$$

意义自明，U 的第 i 列表示 M 的第 i 个特征值对应的特征向量 (这里假设特征值是按顺序排列了)。现在我们需要多大比例的保持方差极大信息，选择一定数量的特征值及其特征向量即可。

“PCA 具有保持子空间拥有最大方差的最优正交变换的特性。然而，当与离散余弦变换相比时，它需要更大的计算需求代价。非线性降维技术相对于 PCA 来说则需要更高的计算要求。”

CCA

典型相关分析:CCA

CCA 寻找两个具有相互关系的随机变量的特征的线性组合，使其表示成的新特征之间具有最大的相关性。可以说是一种保持特征相关性的特征重构。具有降维的作用。其计算过程和 PCA 差不多，首先根据两随机向量 X, Y 计算其互协方差矩阵，然后求解向量 a, b 使得 $\rho = corr(a^{'}X, b^{'}Y)$ 最大，其中 $U = a^{'}X, V = b^{'}Y$ 是第一对典型变量，然后依次求得不相关的典型变量对。而这个问题最后被转化成一个求由协方差阵组合成的某对称矩阵的特征向量问题。

相关代码参考:PyCCA

Multidimensional scaling

多维标度:MDS

代码参考:PyMDS

AutoEncoder

AutoEncoder

从维基上我们看到，自编码是一种无监督式的数据重构方法，其理论比较简单，相应的利用 Tensorflow 或者 Pytorch 实现它也很简单，其扩展方式很多。

现在我们来看看采用概率图模型的自编码方法:Variational autoencoder。这里算了提前进入机器学习概率这一板块了，讲道理，这块是我的弱项。算是提前在这里熟悉概率的一些基本的东西吧。

通俗 VAE此讲解作为第一次阅读，以及后面的彩蛋，都不错。结合入门 VAE该文章小错误比较多，作为入门理解，还是不错的，且不可关注过多细节。入门 2AVE

基础阅读材料:TutorialVAE以及简短的变分推理 Blei, David M. ”Variational Inference.” Lecture from Princeton。

传统图像预处理

滤波,直方图分析,基本射影变换,增强等均可直接调用库Pillow-document。更多方法，比如 Harris Corner,Canny,Sift，视频处理等就需要看 opencv 文档的了：opencv

例子

因为之前看过 Sift，所以这里打算写写对 Sift 的认识。想了下，还是看看这篇已经写好的文章吧。Sift

第二章　机器学习

本章包含了机器学习的经典算法. 经典的机器学习算法有很多库都已经实现，我们没必要所有都去造轮子，我的选择是理解其数学部分，使用现有的库 sklearn，并在实践中分析理论与和实际的差距。假设我们对理论和库的调用都不太熟悉，实际上 sklearn 的 document 本身就是一个很好的学习地方，那里包含了算法的相关参考文献。后面的章节我们首先以这种方式来学习经典机器学习。

开始页面：sklearn-user-guide

4.6 核近似

4.7 Pairwise metrics, Affinities and Kernels

4.8 变换目标值

5 数据导入

6 大数据

以上是 sklearn 指导文档首页的部分目录，现在我们随机选择一些东西学习，比如我这里选择了接下来的四节内容。这只是一个初步的学习，剩下的就是在实践中不断的深化理解实际和理论上的差别，然后再反过来思考理论上的问题。这部分的理论相对简单，但这些优化方法却是人工智能的基础。

Regression

进入一般线性模型，玲琅满目，眼花缭乱。

2.0.1 Logistic regression

逻辑回归是一个二分类概率模型，其很容易扩展到多元情形，它将特征向量映射为一个概率向量，其每个分量表示特征属于其对应标签的概率。模型可表示如下：

$$P^{LR}(W) = C\sum_{i=1}^{n} log(1 + e^{-y_i W^T \mathrm{x_i}}) + 1/2W^T W \tag{2.1}$$

其中 $\{\mathrm{x_i}, y_i\}_{i=1}^{n}$ 表示数据以及其标签，$\mathrm{x_i} \in R^m, y_i \in \{1, -1\}.C > 0, W$ 是要学习的参数。

给定数据及其标签，我们可以用如下公式来表示条件概率。

$$P_W(y = \pm 1|\mathrm{x}) = \frac{1}{1 + e^{-yW^T\mathrm{x}}} \tag{2.2}$$

根据极大似然原理，我们很容易由 (2.2) 得到 (2.1)，如果我们去掉 (2.1) 的 $1/2W^T W$，这个多余的东西其实就是正则项，用来限制参数 W，防止过拟合的技巧。这个后面详说。在实际情况中，往往需要很多额外技巧来使模型更加实用。

现在的问题是如何得到模型参数 W, C?

答案：Coordinate descent approch, quasi-Newton method,iterative scaling method, exponetiatel gradient ... 如果你学过数值分析的话，你会觉得很多似曾相识，如果你学过凸分析的话，你会觉得很亲切，随着学习的深入，我们会逐渐建立更清晰的理论框架。

现在不妨将视角转向 SVM。

Linear regression Logisticregression

补充

关于多分类的推广形式可参看：Softmax 回归. 摘录一段算法对比说明：

Softmax 回归 vs. k 个二元分类器

机器学习更多的时候是要面向实际问题的，每一个算法在面对实际问题时都会有很多其他问题，包括对算法本身的理解程度，以及对实际问题的适应情况等。

如果你在开发一个音乐分类的应用，需要对 k 种类型的音乐进行识别，那么是选择使用 softmax 分类器呢，还是使用 logistic 回归算法建立 k 个独立的二元分类器呢？这一选择取决于你的类别之间是否互斥，例如，如果你有四个类别的音乐，分别为：古典音乐、乡村音乐、摇滚乐和爵士乐，那么你可以假设每个训练样本只会被打上一个标签（即：一首歌只能属于这四种音乐类型的其中一种），此时你应该使用类别数 k = 4 的 softmax 回归。（如果在你的数据集中，有的歌曲不属于以上四类的其中任何一类，那么你可以添加一个“其他类”，并将类别数 k 设为 5。）如果你的四个类别如下：人声音乐、舞曲、影视原声、流行歌曲，那么这些类别之间并不是互斥的。例如：一首歌曲可以来源于影视原声，同时也包含人声。这种情况下，使用 4 个二分类的 logistic 回归分类器更为合适。这样，对于每个新的音乐作品，我们的算法可以分别判断它是否属于各个类别。现在我们来看一个计算视觉领域的例子，你的任务是将图像分到三个不同类别中。(i) 假设这三个类别分别是：室内场景、户外城区场景、户外荒野场景。你会使用 sofmax 回归还是 3 个 logistic 回归分类器呢？(ii) 现在假设这三个类别分别是室内场景、黑白图片、包含人物的图片，你又会选择 softmax 回归还是多个 logistic 回归分类器呢？在第一个例子中，三个类别是互斥的，因此更适于选择 softmax 回归分类器。而在第二个例子中，建立三个独立的 logistic 回归分类器更加合适。

补充： 为什么 LR 模型使用 Sigmoid 函数?

一 根据概率假设建立模型推到出来如此，最大熵原则

二 函数本身的性质符合我们设计模型时需要考虑的一些东西比如将线性分类器映射到 0-1, 且单调上升。

三 根据物理的波耳兹曼能量方程等解释

softmax 是最大熵模型的结果，玻尔兹曼分布也是热力学熵最大（微观状态等概率）的结果，所以它们具有相同的结果就是理所当然的了

softmax 算法为什么采用 softmax function 作为每一个类别的概率?

本质上，linear，Logistic，Softmax 都是一个东西推导出来的。就是广

义线性模型。这些分布之所以长成这个样子，是因为我们对 y 进行了假设。当 y 是正太分布———>linear model 当 y 是两点分布———>Logistic model 当 y 是多项式分布———>Softmax 只要 y 的分布是指数分布族的（还有若干假设），都可以用一种通用的方法推导出 h(x)。所以你去了解一下广义线性模型，他推导出来就是这个样子的。

支持向量机

SVM，一个二分类线性模型，简单的说，就是我们高中遇见的线性规划问题的推广。我们知道直线 $y = kx + b$ 将平面 xy 分成两部分，其实也就是两类，一类在“上面”，一类在“下面”。现在我们的情况只是在维度上增加了，也就是寻找一个超平面 $y = Wx + b$ 能将数据分类出来。模型可表示如下：

$$P^{SVM}(W) = C\sum_{i=1}^{n} max(1 - y_i W^T \mathrm{x_i}, 0) + 1/2W^T W \tag{2.3}$$

很明显超平面是依赖训练数据的。从文档上看，该分类其的好处有：

- 高维空间上比较高效。
- 对特征维度大于样本量时，仍然有效 (大过多时，需要适当的选择核函数和正则项)。
- 用部分数据来得到决策函数，也即支持向量，能减少内存。

上面只是最简单的情形，多分类，多元回归呢?

我们先来看看文档里的情况，对于多分类，从文档里我们了解到两种方法：SVC 的“one-against-one”，n 类标签构建 $\binom{n}{2}$ 个分类器;LinearSVC 的“one-vs-the-rest”，训练 n 个模型。

对于回归问题，其对应的模块名叫 SVR。自行查看即可。现在的问题是：怎么从分类模型过度到回归模型呢？完整想出来，似乎还是有点难度，但是我们看到网页所给参看文献SVR，好了，又到了真正学习的时候了。

此段讲理论，以及代码分析。

然而实际上我们对多分类问题的处理方式还是失望的，我们并不想重复二分类模型，针对这个问题，表明我们该看看 1.12 节Multiclass-Multilabel了。

Maximum Entropy Softmax 函数与交叉熵

蹦，问题又来了，所谓超平面，直观上看，毕竟是个“平”的。实际问题中，数据往往需要用一个弯曲的面才能将其较好的分类出来，这时怎么办呢？自然的我们有两种想法，一种直接把超曲面算出来，但这样不太好，考虑到曲面的表示方式，能控制的范围太小了，而实际变化范围太大; 另一种

方法就是保持超平面不变，直接映射输入数据，使其能被平面分割。这就是所谓的核技巧。

核技巧讲完了。

现在我们来看看以上模型该如何学习参数。

随机选择一个 Reference:Dual coordinate descent for LR and ME

如果你想自己造 SVM 的轮子，单从 sklearn 里面是不可能的，于是我们将眼光着眼向其他地方，课程 cs229 就是一个很好的地方，cs229 里面关于 SVM 部分写的很好，也有相应的 SMO 算法详解（一个简化版本的 SVM 对偶问题的优化算法）。

补充：

SVM 和 logistic 回归分别在什么情况下使用？orangeprince 回答：两种方法都是常见的分类算法，从目标函数来看，区别在于逻辑回归采用的是 logistical loss，svm 采用的是 hinge loss。这两个损失函数的目的都是增加对分类影响较大的数据点的权重，减少与分类关系较小的数据点的权重。SVM 的处理方法是只考虑 support vectors，也就是和分类最相关的少数点，去学习分类器。而逻辑回归通过非线性映射，大大减小了离分类平面较远的点的权重，相对提升了与分类最相关的数据点的权重。两者的根本目的都是一样的。此外，根据需要，两个方法都可以增加不同的正则化项，如 l1,l2 等等。所以在很多实验中，两种算法的结果是很接近的。但是逻辑回归相对来说模型更简单，好理解，实现起来，特别是大规模线性分类时比较方便。而 SVM 的理解和优化相对来说复杂一些。但是 SVM 的理论基础更加牢固，有一套结构化风险最小化的理论基础，虽然一般使用的人不太会去关注。还有很重要的一点，SVM 转化为对偶问题后，分类只需要计算与少数几个支持向量的距离，这个在进行复杂核函数计算时优势很明显，能够大大简化模型和计算量

Andrew Ng 的建议：n 是 feature 的数量 m 是样本数
1、如果 n 相对于 m 来说很大，则使用 LR 算法或者不带核函数的 SVM（线性分类）n 远大于 m，n=10000，m=10-1000
2、如果 n 很小，m 的数量适中 n=1-1000，m=10-10000）使用带有核函数的 SVM 算法
3、如果 n 很小，m 很大（n=1-1000，m=50000+）增加更多的 feature 然后使用 LR 算法或者不带核函数的 SVMLR 和不带核函数的 SVM 比较类似。

Adaboost

高斯混合模型

高斯混合模型属于生成模型，也即数据集没有标签，我们需要学习的是数据集本身该有的结构，换句话就是数据的概率分布情况。而根据任何概率分布都可以用高斯混合分布去逼近，所以学习数据概率分布，就可以直接学习其概率混合分布了。这里主要涉及到经典的 EM 算法，网络资源已经有很好的了，就不再赘述。从Why GMM里我们看到，高斯混合分布逼近不一定就是最佳逼近，但好算始终是真的，不过高维情况收敛很慢，因为这里面要进行采样，经典的算法是 Gibbs 采样，也即你要用到马尔科夫过程，收敛速率慢也就可以理解了。

一个例子：假设你有一堆有男有女的身高数据，那么怎么预测新来的数据是男生还是女生呢？这时你就要从已知数据里学习男女身高的混合高斯分布，也即你只需要学习到两个平均数，两个方差以及对应的模型占比即可。

后面时机成熟会在 github 上附上高维版本的代码。

第三章　理论知识

列表

暂时略去，看情况选择补充与否。

第四章　深度学习

首先推荐的书籍:neuralnetworks 完全根据 numpy 和 python 从最基本的 bp 到神经网络模型的实现。pytorch-tutorial pythonic 型的 pytorch 教程，包含了基本的回归和全连接层网络，cnn,rnn,resnet,cnn-rnn,dcgan,vae 等，力推。关于以上内容的基本概念可以看看斯坦福教程UFLDL 文档

相关比较 toy 的代码，可以参看我的 github。

神经网络

卷积神经网络

CNN 流程经验 特别将其应用场景列举出来：

1. 图像识别与检索

2. 人脸识别

3. 性别/年龄/情绪识别

4. 物体检测

用 pytorch 来实现 cnn 是很简单的，关于上面的具体应用部分，在实际情况中是很复杂的，cnn 只是充当了一个很小的角色，比如特征提取，经典的结构有 vgg, resnet 等。物体检测方面我将在论文阅读章节中讲讲对 maskrcnn 的理解。

CNN、RNN、DNN 的内部结构区别可参看知呼回答情况：C(R,D)NN 结构区别

RNN

以下这些连接，更多的是以后要了解时 (如果做图像标注)，也许可以返回来看看 (被筛选)。

总体了解下 RNN 的应用场景：RNN 应用 理论上只要问题可以建模成序列到序列的映射均可以采用 RNN 进行求解，例如：

1. 命名实体识别问题 NER
2. 词级别的情感分析
3. 问答系统
4. 机器翻译
5. 图片添加注释
6. 文本摘要
7. RNN/LSTM 应用论文列表

再来经历经历从入门到放弃的过程。风萧易水寒的 RNN 介绍，包含了文本生成的代码实现。风萧易水寒 RNN

李飞飞的学生写的关于 RNN 的文章 rnn-effectiveness

随机一篇关于 LSTM 的理解文章，这方面其实可以看台湾李宏毅教授的视频。LSTM-理解

RNN 以及 LSTM 的介绍和公式梳理 LSTM 公式梳理

Gaussian Processes

斗大的熊猫的博客，也包含了 tensorflow 教程，内容都很实用。RNN 生成古诗词

GAN

目前看来，深度学习或者人工智能领域知识更新太快，真正属于知识的或许不多。我的感受是在短时间了解了各个方向的情况后，应该将范围缩小，集中深入学习点东西，比如物体分割以及识别。有公司实际项目更好。所以此节以及后面各节有机会补充。更多的是在论文阅读章节进行补充。

4.0.1 DCGAN

How To Train a GAN 其实这个算法生成的照片效果不好，后面 Adobe 公式给出了能生成高质量图片的算法Adobe-GAN，但是训练成本太高，所以静待时光流逝吧。

DQN

A3C

DDPG

第五章　传统智能

包含了传统智能算法.

AG

有机会补充。

第六章　论文阅读

其实也没什么。

GAN

Mask R-CNN

我们知道 Mask R-CNN 是在一系列的工作之上的工作，所以看 paper 的时候必然会连带的看上 4,5 篇文章。包括 r-cnn,faster-rcnn,fpn,vgg,resnet 等。相应的对整个框架的细节理解难度就会增加，所以程序的辅助作用还是必要的，最终会以以下两个版本的复现为准进行细致的分析。rgb-py-faster-rcnn pytorch-faster-rcnn

6.0.1　Faster R-CNN

首先来看看 faster-rcnn，以下内容几乎来自faster-rcnn 解析。

Faster R-CNN 的思想和框架

Faster R-CNN 可以简单地看做“区域生成网络 RPNs + Fast R-CNN”的系统，用区域生成网络代替 FastR-CNN 中的 Selective Search 方法。Faster R-CNN 这篇论文着重解决了这个系统中的三个问题：

1. 如何设计区域生成网络;

2. 如何训练区域生成网络;

3. 如何让区域生成网络和 Fast RCNN 网络共享特征提取网络。

Faster-R-CNN 算法由两大模块组成：

1. RPN 候选框提取模块;

2. Fast R-CNN 检测模块。

其中，RPN 是全卷积神经网络，用于提取候选框;Fast R-CNN 基于 RPN 提取的 Proposal 检测并识别 Proposal 中的目标。

RPN

背景

目前最先进的目标检测网络需要先用区域建议算法推测出目标位置，像 SPPnet 和 Fast R-CNN 这些网络虽然已经减少了检测网络运行的时间，但是计算区域建议依然耗时很大。所以，在这样的瓶颈下，RGB 和 Kaiming He 一帮人将 Region Proposal 也交给 CNN 来做，这才提出了 Region Propsal Network 区域建议网络来提取检测区域，它和整个检测网络共享全图发卷积特征，使得区域建议几乎不花时间。

具体操作

首先将原始输入的 image 经过一些基本处理，比如固定大小 (600*1000)，根据选定的大小进行 Anchor 设定，Anchor 基本要覆盖输入的图片 (这里 9 个)，此时需选择特征映射网络比如 ZF，Vgg16 或者 ResNet 从固定大小的输入图片中得到较小的特征图片 (40*60，共 256 张，再合成一个 256 维的向量：表示位置?，后面根据这个向量进行类别判断以及位置回归，下面将会仔细说明)，同时使用 3×3 的窗口在特征图上滑动，3×3 的中心对应原图 (re-scale) 上的位置，将该点作为 Anchor 的中心点（这里有个问题：特征图上的点是怎么对应到原图上的?），于是产生了 40×60×9 个 anchor boxes，去除与边界相交的 anchor boxes 剩下约 6k 个 achor boxes，然后使用非最大抑制 (NMS,non-maximum suppression) 将 $IOU > 0.7$ 的区域全部合并，剩下约 2k 个 anchor boxes(同理，在最终检测端，可以设置概率大于某阈值且 IOU 大于某阈值的预测框采用 NMS 方法进行合并)。NMS 不会影响最终的检测准确率，但是大幅地减少了建议框的数量。NMS 之后，我们用建议区域中的 top-N 个来检测（即排过序后取 N 个）。

关于上面的仔细说明：

1. 将每个特征图的位置编码成一个特征向量（256dfor ZF and 512d for VGG）。

2. 对每一个位置输出一个 objectness score 和 regressed bounds for k 个 region proposal，即在每个卷积映射位置输出这个位置上多种尺度（3 种）和长宽比（3 种）的 k 个（3*3=9）区域建议的物体得分和回归边界。即：

分类层（cls_score）输出每一个位置上，9 个 anchor 属于前景和背景的概率。

窗口回归层（$bbox\_pred$）输出每一个位置上，9 个 anchor 对应窗口应该平移缩放的参数 (x, y, w, h)。

RPN 网络的输入可以是任意大小（但还是有最小分辨率要求的，例如 VGG 是 228*228）的图片。如果用 VGG16 进行特征提取，那么 RPN 网络的组成形式可以表示为 VGG16+RPN。

注意：在特征图上扫描时，是将 3×3 的窗口映射到低维向量（表位置，256 维），这里应该是 40×60×256 中的一个 3×3 区域即 3×3×256 映射到 1×256 的位置向量。将这个低维向量送入到两个全连接层，即 bbox 回归层（reg）和 box 分类层（cls）。对每个位置进行前景背景概率预测（$p = (p_0, p_1, \ldots, p_k)$ 共 $k+1$ 维，SoftMax 即可完成，这里每个 Anchor 对应原图的 Proposal 的标签已经由 IOU 给出。。）以及对应原图的窗口平移缩放参数预测（(x, y, w, t)）。当然实际训练的时候，不是 40×60×9 全部都如此，而是预先利用 IOU 将 Proposal 筛选到极小，再进行这一步。sliding window 的处理方式保证 reg-layer 和 cls-layer 关联了 conv5-3 的全部特征空间。另外：并没有显式地提取任何候选窗口，完全使用网络自身完成判断和修正。

我们的最终目标是和 Fast R-CNN 目标检测网络共享计算。

学习区域建议损失函数

A. 标签分类规定：

论文中说明了，满足如下条件之一即可判断为正标签

1. 与 GT 包围盒最高的 IOU 重叠的 anchor(也许不到 0.7)

2. 与任意 GT 包围盒的 IOU 大于 0.7 的 anchor

负标签：与所有的 GT 包围和的 IOU 小于 0.3 的 anchor。

对于既不是正标签也不是负标签的 anchor，以及跨越图像边界的 anchor 我们给予舍弃，因为其对训练目标是没有任何作用的。

B. 多任务损失：
也就是在一个网络里面针对多个任务而提出一个综合性损失函数。在这里 RPN 的最终损失函数可以表示如下：

$$L(\{p_i\}, \{u_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_1, p^*) + \lambda \frac{1}{N_{reg}} \sum_i p^* L_{reg}(t_i, t_i^*)$$

p_i为 anchor 预测为目标的概率;

GT标签：$p_i^* = \begin{cases} 0 & negative\ label \\ 1 & positive\ label \end{cases}$；

$t_i = \{t_x, t_y, t_w, t_h\}$是一个向量，表示预测的 bounding box 包围盒的 4 个参数化坐标;

t_i^*是与 positive anchor 对应的 ground truth 包围盒的坐标向量;

$L_{cls}(p_i, p_i^*)$是两个类别（目标 vs.非目标）的对数损失：

$$L_{cls}(p_i, p_i^*) = -\log\left[p_i^* p_i + (1 - p_i^*)(1 - p_i)\right]$$

$L_{reg}(t_i, t_i^*)$是回归损失，用 $L_{reg}(t_i, t_i^*) = R(t_i - t_i^*)$ 来计算，R 是 smooth L1 函数。

$p_i^* L_{reg}$这一项意味着只有前景 anchor（$p_i^* = 1$）才有回归损失，其他情况就没有（$p_i^* = 0$）。cls 层和 reg 层的输出分别由$\{p_i\}$和$\{u_i\}$组成，这两项分别由N_{cls}和N_{reg}以及一个平衡权重λ归一化（早期实现及公开的代码中，$\lambda = 10$，cls 项的归一化值为 mini-batch 的大小，即$N_{cls} = 256$，reg 项的归一化值为 anchor 位置的数量，即N_{reg}~2400（$40 * 60$），这样 cls 和 reg 项差不多是等权重的。

Boundingbox 回归

函数优化目标:

$$W_* = \underset{\hat{W}_*}{\operatorname{argmin}} \sum_i^N (t_*^i - \hat{W}_*^T \Phi_5(P^i))^2 + \lambda ||\hat{W}_*||^2$$

其中 $\Phi_5(P^i)$ 为输入的 256 维特征，$*$ 表示 x, y, w, h，每个映射得到一个数值 (变化量)，比如 $\hat{t_x}^*$，然后根据关系返回计算预测值 x。具体关系如下:

$$\begin{aligned}
t_x &= (x - x_a)/w_a, \quad t_y = (y - y_a)/h_a, \\
t_w &= \log(w/w_a), \quad t_h = \log(h/h_a), \\
t_x^* &= (x^* - x_a)/w_a, \quad t_y^* = (y^* - y_a)/h_a, \\
t_w^* &= \log(w^*/w_a), \quad t_h^* = \log(h^*/h_a),
\end{aligned}$$

其中:

x，y，w，h为 box 的中心坐标、宽、高；

$$\begin{cases} x：\text{predicted box(就是 Proposal)} \\ x_a：\text{anchor box（}40*60*9\text{ 个）} \\ x^*：\text{Ground Truth(正确标定的 GT)} \end{cases}$$

注意 1：只有当 Proposal 和 Ground Truth 比较接近时（线性问题），我们才能将其作为训练样本训练我们的线性回归模型，否则会导致训练的回归模型不 work（当 Proposal 跟 GT 离得较远，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）。这个也是 G-CNN: an Iterative Grid Based Object Detector 多次迭代实现目标准确定位的关键 (当相差很大时，算出的差值 t_x 抖动厉害，不是线性关系？)。

注意 2：计算 regression loss 需要三组信息：

1) 预测框，即 RPN 网络测出的 proposa；

2) 锚点 anchor box：之前的 9 个 anchor 对应 9 个不同尺度和长宽比的 anchorbox；

3)GroundTruth：标定的框。

问题：

训练 RPNs

采样

初始化

新增的 2 层参数用均值为 0，标准差为 0.01 的高斯分布来进行初始化，其余层（都是共享的卷积层，与 VGG 共有的层）参数用 ImageNet 分类预训练模型来初始化。

参数设置

在 PASCAL 数据集上：

前 60k 个 mini-batch 进行迭代，学习率设为 0.001；

后 20k 个 mini-batch 进行迭代，学习率设为 0.0001；

设置动量 momentum=0.9，权重衰减 weightdecay=0.0005。

学习细节

RPN 与 Fast R-CNN 特征共享

我们已经描述了如何为生成区域建议训练网络，而没有考虑基于区域的目标检测 CNN 如何利用这些建议框。对于检测网络，我们采用 Fast R-CNN，现在描述一种算法，学习由 RPN 和 Fast R-CNN 之间共享的卷积层。

RPN 和 Fast R-CNN 都是独立训练的，要用不同方式修改它们的卷积层。因此需要开发一种允许两个网络间共享卷积层的技术，而不是分别学习两个网络。注意到这不是仅仅定义一个包含了 RPN 和 Fast R-CNN 的单独网络，然后用反向传播联合优化它那么简单。原因是 Fast R-CNN 训练依赖于固定的目标建议框，而且并不清楚当同时改变建议机制时，学习 Fast R-CNN 会不会收敛。

RPN 在提取得到 proposals 后，作者选择使用 Fast-R-CNN 实现最终目标的检测和识别。RPN 和 Fast-R-CNN 共用了 13 个 VGG 的卷积层，显然将这两个网络完全孤立训练不是明智的选择，作者采用交替训练（Alternating training）阶段卷积层特征共享：

第一步，我们依上述训练 RPN，该网络用 ImageNet 预训练的模型初始化，并端到端微调用于区域建议任务；

第二步，我们利用第一步的 RPN 生成的建议框，由 Fast R-CNN 训练一个单独的检测网络，这个检测网络同样是由 ImageNet 预训练的模型初始化的，这时候两个网络还没有共享卷积层；

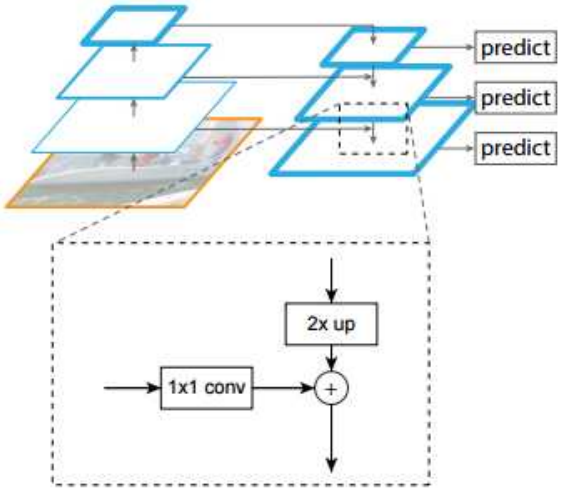
第三步，我们用检测网络初始化 RPN 训练，但我们固定共享的卷积层，并且只微调 RPN 独有的层，现在两个网络共享卷积层了；

第四步，保持共享的卷积层固定，微调 Fast R-CNN 的 fc 层。这样，两个网络共享相同的卷积层，构成一个统一的网络。

注意：第一次迭代时，用 ImageNet 得到的模型初始化 RPN 和 Fast-R-CNN 中卷积层的参数；从第二次迭代开始，训练 RPN 时，用 Fast-R-CNN 的共享卷积层参数初始化 RPN 中的共享卷积层参数，然后只 Fine-tune 不共享的卷积层和其他层的相应参数。训练 Fast-RCNN 时，保持其与 RPN 共享的卷积层参数不变，只 Fine-tune 不共享的层对应的参数。这样就可以实现两个网络卷积层特征共享训练。

6.0.2 FPN

论文链接：FPN-paper 这篇文章是对 RPN 网络的输入加强，准确点说就是原先 RPN 的输入是 VGG 等网络的某一特征层，也就是某一单一尺度，现在的思路基本来源于传统图像处理的方式，做做不同尺度的特征，再将其融合。也就是图像金字塔。大体思路根据论文中的结构图即可知一二。



从上图可以看到，算法结构有三个部分，自底向上，横向以及自顶向下。自底向上输出不变特征，自顶向下采用 upsampling，横向连接则将上采样的结果和自底向上生成的相同大小的特征图进行融合。在融合之后还会再采用 3*3 的卷积核对每个融合结果进行卷积，目的是消除上采样的混叠效应 (aliasing effect)。这样每层都将有个融合特征，然后每层做独立预测。这里有区别于其他文章的处理方式，比如 Single Shot Detector, 谁好谁坏，具体情况分析吧。

剩下的就是将 FPN 网络，这种多层多尺度预测形式，放到 RPN 网络中去了，以前是单尺度预测。也就是平行的增加了几个 scale 层，改变对应 scale 的 anchor 即可。

最后，这种算法的改进地方肯定很多。具体得用实验来分析了。做不了实验的先逃吧。

6.0.3 Focal Loss

论文链接：Focal Loss。看完论文再看看这个，就更加清晰了。中文 这篇文章比较简单，但是完整的网络 RetinaNet 我还是复现不出来 (弱)。作者指明了 Single stage detector(SSD) 不好的原因是极度不平衡的正负样本比例以及梯度被易检测样本影响所致。所以最后也就是一个公式的事 (样本不平衡反映到损失函数的贡献量，导致训练方向不一定对，原始的损失函数上，易分类样本对总体损失的贡献不一定合理)。对损失函数交叉熵的改进。怎么能做出这样的工作呢？我想必要的是对问题本身以及理论的深刻认识吧。做到这两点本身是需要条件的。最后看看公式吧。对新添的两个地方的具体分析论文中已经说的很清楚了，不再赘述。

由$CE(p_t) = -log(p_t)$,改成$FL(p_t) = -\alpha_t(1-p_t)^\gamma log(p_t)$

问题：有其他问题对损失函数进行修改么？此扩展版本的交叉熵可以扩展到多分类中去么？

6.0.4 VGG

VGG 的结构相对简单明了，见下图。它是用来学习特征的，但 VGG 结构好在哪里，中间层的数字和输入图片大小的关系怎样，不同 VGG 之间的区别是什么，它能作为一种基础的特征表达网络么，又如何对其进行推广？这些问题等以后来解答吧。

Table 1: **ConvNet configurations** (shown in columns). The depth of the configurations increases from the left (A) to the right (E), as more layers are added (the added layers are shown in bold). The convolutional layer parameters are denoted as "conv⟨receptive field size⟩-⟨number of channels⟩". The ReLU activation function is not shown for brevity.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Table 2: **Number of parameters** (in millions).

Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

6.0.5 Mask-RCNN

Mask-RCNN 看了这么多，就为了一个 Mask-RCNN，虽然等会你就知道了 Mask-RCNN 的主要内容，但是细节还是很多不清楚 (更弱)。

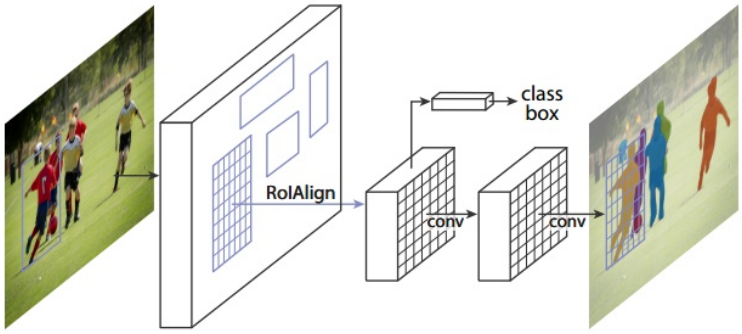
Mask-RCNN 是在 Faster-RCNN 的网络框架上做的改进，主要改进有三点：

1. 基础特征网络的增强：ResNeXt-101+FPN

2. 分割 loss 的改进：softmax->sigmod

3. 池化层的改进：RoIPooling->RoIAlign

回看 Focal Loss 章节，我提出了一个问题，对损失函数的修改例子，这里就是一个例子。



现对上面的改进作进一步的理解。

Mask R-CNN：两步过程

一,RPN: 给出候选区域的 bbox

二, 分支一：各个候选框中进行分类和 bbox offset。分支二：对每个 RoI 输出 binary mask

损失函数：$L = L\_cls + L\_box + L\_mask$ mask 分支对于每个 RoI 有 Km^2 维度的输出。K 个（类别数）分辨率为 m*m 的二值 mask。因此作者利用了 a per-pixel sigmoid，并且定义 L_mask 为平均二值交叉熵损失. 对于一个属于第 k 个类别的 RoI，L_mask 仅仅考虑第 k 个 mask（其他的掩模输入不会贡献到损失函数中）。这样的定义会允许对每个类别都会生成掩模，并且不会存在类间竞争。

RoIAlign: RoI Pooling 在 Pooling 时可能会有 misalignment。解决方法：参考 Spatial Transformer Networks (NIPS2015), 使用双线性插值，再做聚合。

上面其实也就是论文的中文简写，真正隐含的更多细节，还是有待慢慢理解。最后附 Tensorflow 版本的代码 MaskRCNN-tensorflow。

问题：Focal Loss 用在 Mask-RCNN 上效果会怎样?