CS231n课程笔记翻译：线性分类笔记（下）

杜客
求索

已关注

276 人赞同了该文章

译者注：本文智能单元首发，译自斯坦福CS231n课程笔记Linear Classification Note，课程教师Andrej Karpathy授权翻译。本篇教程由杜客翻译完成，堃堃进行校对修改。译文含公式和代码，建议PC端阅读。

原文如下

内容列表：

Softmax分类器

SVM是最常用的两个分类器之一，而另一个就是**Softmax分类器**，它的损失函数与SVM的损失函数不同。对于学习过二元逻辑回归分类器的读者来说，Softmax分类器就可以理解为逻辑回归分类器面对多个分类的一般化归纳。SVM将输出 $f(x_i, W)$ 作为每个分类的评分（因为无定标，所以难以直接解释）。与SVM不同，Softmax的输出（归一化的分类概率）更加直观，并且从概率上可以解释，这一点后文会讨论。在Softmax分类器中，函数映射 $f(x_i; W) = Wx_i$ 保持不变，但将这些评分值视为每个分类的未归一化的对数概率，并且将*折叶损失（hinge loss）*替换为**交叉熵损失（cross-entropy loss）**。公式如下：

$$Li = -log(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}) \text{ 或等价的 } L_i = -f_{y_i} + log(\sum_j e^{f_j})$$

在上式中，使用 f_j 来表示分类评分向量 f 中的第j个元素。和之前一样，整个数据集的损失值是数据集中所有样本数据的损失值 L_i 的均值与正则化损失 $R(W)$ 之和。其中函数 $f_j(z) = \frac{e^{z_j}}{\sum_k e^{z_k}}$ 被称作**softmax 函数**：其输入值是一个向量，向量中元素为任意实数的评分值（z中的），函数对其进行压缩，输出一个向量，其中每个元素值在0到1之间，且所有元素之和为1。所以，包含softmax函数的完整交叉熵损失看起唬人，实际上还是比较容易理解的。

信息理论视角：在“真实”分布 p 和估计分布 q 之间的*交叉熵*定义如下：

$$H(p,q) = -\sum_x p(x) log q(x)$$

因此，Softmax分类器所做的就是最小化在估计分类概率（就是上面的 $e^{f_{y_i}} / \sum_j e^{f_j}$ ）和“真实”分布之间的交叉熵，在这个解释中，“真实”分布就是所有概率密度都分布在正确的类别上（比如：$p = [0, \ldots 1, \ldots, 0]$ 中在 y_i 的位置就有一个单独的1）。还有，既然交叉熵可以写成熵和相对熵（Kullback-Leibler divergence） $H(p,q) = H(p) + D_{KL}(p||q)$，并且delta函数 p 的熵是0，那么就能等价的看做是对两个分布之间的相对熵做最小化操作。换句话说，交叉熵损失函数“想要”预测分布的所有概率密度都在正确分类上。

***译者注**：Kullback-Leibler差异（Kullback-Leibler Divergence）也叫做相对熵（Relative Entropy），它衡量的是相同事件空间里的两个概率分布的差异情况。*

概率论解释：先看下面的公式：

$$P(y_i|x_i, W) = \frac{e^{f_{y_i}}}{\sum_j e^{f_j}}$$

可以解释为是给定图像数据 x_i，以 W 为参数，分配给正确分类标签 y_i 的归一化概率。为了理解这点，请回忆一下Softmax分类器将输出向量 f 中的评分值解释为没有归一化的对数概率。那么以这些数值做指数函数的幂就得到了没有归一化的概率，而除法操作则对数据进行了归一化处理，使得这些概率的和为1。从概率论的角度来理解，我们就是在最小化正确分类的负对数概率，这可以看做是在进行*最大似然估计*（MLE）。该解释的另一个好处是，损失函数中的正则化部分 $R(W)$ 可以被看做是权重矩阵 W 的高斯先验，这里进行的是最大后验估计（MAP）而不是最大似然估计。提及这些解释只是为了让读者形成直观的印象，具体细节就超过本课程范围了。

实操事项：数值稳定。编程实现softmax函数计算的时候，中间项 $e^{f_{y_i}}$ 和 $\sum_j e^{f_j}$ 因为存在指数函数，所以数值可能非常大。除以大数值可能导致数值计算的不稳定，所以学会使用归一化技巧非常重要。如果在分式的分子和分母都乘以一个常数 C，并把它变换到求和之中，就能得到一个从数学上等价的公式：

$$\frac{e^{f_{y_i}}}{\sum_j e^{f_j}} = \frac{Ce^{f_{y_i}}}{C\sum_j e^{f_j}} = \frac{e^{f_{y_i}+logC}}{\sum_j e^{f_j+logC}}$$

C的值可自由选择，不会影响计算结果，通过使用这个技巧可以提高计算中的数值稳定性。通常将C设为$logC = -max_j f_j$。该技巧简单地说，就是应该将向量f中的数值进行平移，使得最大值为0。代码实现如下：

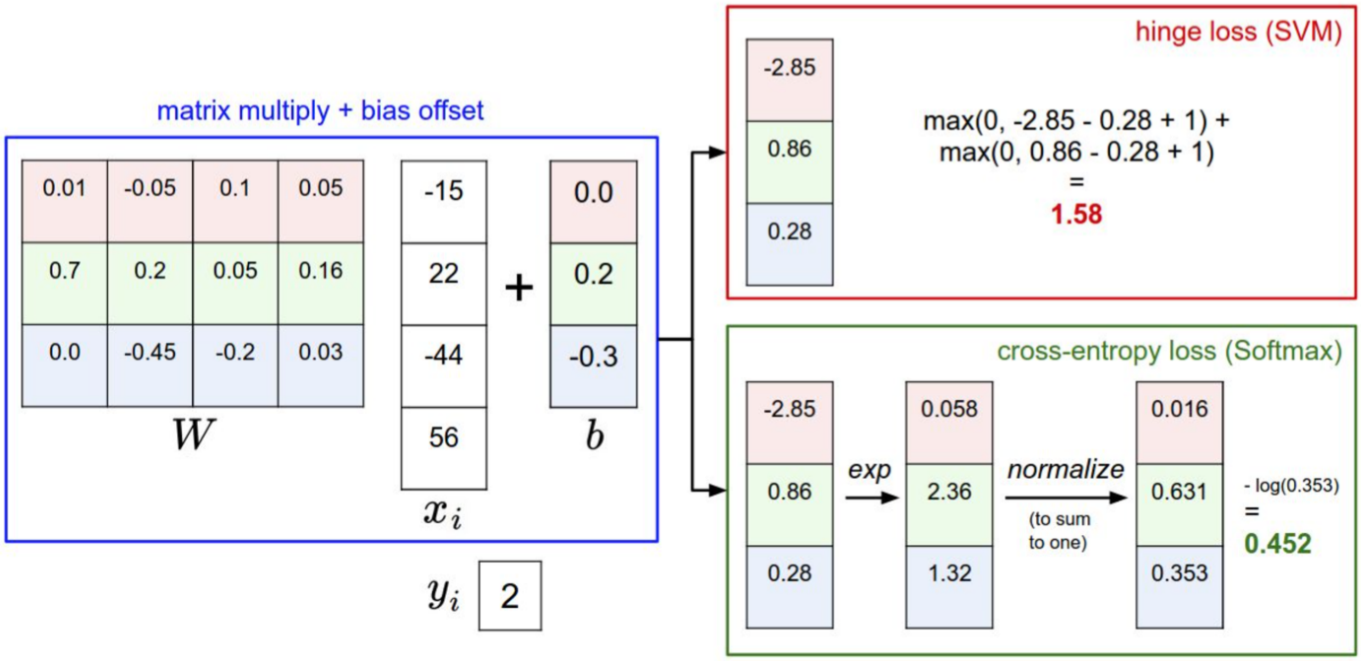
```
f = np.array([123, 456, 789]) # 例子中有3个分类，每个评分的数值都很大
p = np.exp(f) / np.sum(np.exp(f)) # 不妙：数值问题，可能导致数值爆炸

# 那么将f中的值平移到最大值为0：
f -= np.max(f) # f becomes [-666, -333, 0]
p = np.exp(f) / np.sum(np.exp(f)) # 现在OK了，将给出正确结果
```

让人迷惑的命名规则：精确地说，SVM分类器使用的是*折叶损失（hinge loss）*，有时候又被称为*最大边界损失（max-margin loss）*。Softmax分类器使用的是*交叉熵损失（corss-entropy loss）*。Softmax分类器的命名是从*softmax函数*那里得来的，softmax函数将原始分类评分变成正的归一化数值，所有数值和为1，这样处理后交叉熵损失才能应用。注意从技术上说“softmax损失（softmax loss）”是没有意义的，因为softmax只是一个压缩数值的函数。但是在这个说法常常被用来做简称。

SVM和Softmax的比较

下图有助于区分这 Softmax和SVM这两种分类器：



针对一个数据点，SVM和Softmax分类器的不同处理方式的例子。两个分类器都计算了同样的分值向量**f**（本节中是通过矩阵乘来实现）。不同之处在于对**f**中分值的解释：SVM分类器将它们看做是分类评分，它的损失函数鼓励正确的分类（本例中是蓝色的类别2）的分值比其他分类的分值高出至少一个边界值。Softmax分类器将这些数值看做是每个分类没有归一化的**对数概率**，鼓励正确分类的归一化的对数概率变高，其余的变低。SVM的最终的损失值是1.58，Softmax的最终的损失值是0.452，但要注意这两个数值没有可比性。只在给定同样数据，在同样的分类器的损失值计算中，它们才有意义。

Softmax分类器为每个分类提供了“可能性”：SVM的计算是无标定的，而且难以针对所有分类的评分值给出直观解释。Softmax分类器则不同，它允许我们计算出对于所有分类标签的可能性。举个例子，针对给出的图像，SVM分类器可能给你的是一个[12.5, 0.6, -23.0]对应分类“猫”，“狗”，“船”。而softmax分类器可以计算出这三个标签的”可能性“是[0.9, 0.09, 0.01]，这就让你能看出对于不同分类准确性的把握。为什么我们要在”可能性“上面打引号呢？这是因为可能性分布的集中或离散程度是由正则化参数λ直接决定的，λ是你能直接控制的一个输入参数。举个例子，假设3个分类的原始分数是[1, -2, 0]，那么softmax函数就会计算：

$$[1, -2, 0] \rightarrow [e^1, e^{-2}, e^0] = [2.71, 0.14, 1] \rightarrow [0.7, 0.04, 0.26]$$

现在，如果正则化参数λ更大，那么权重W就会被惩罚的更多，然后他的权重数值就会更小。这样算出来的分数也会更小，假设小了一半吧[0.5, -1, 0]，那么softmax函数的计算就是：

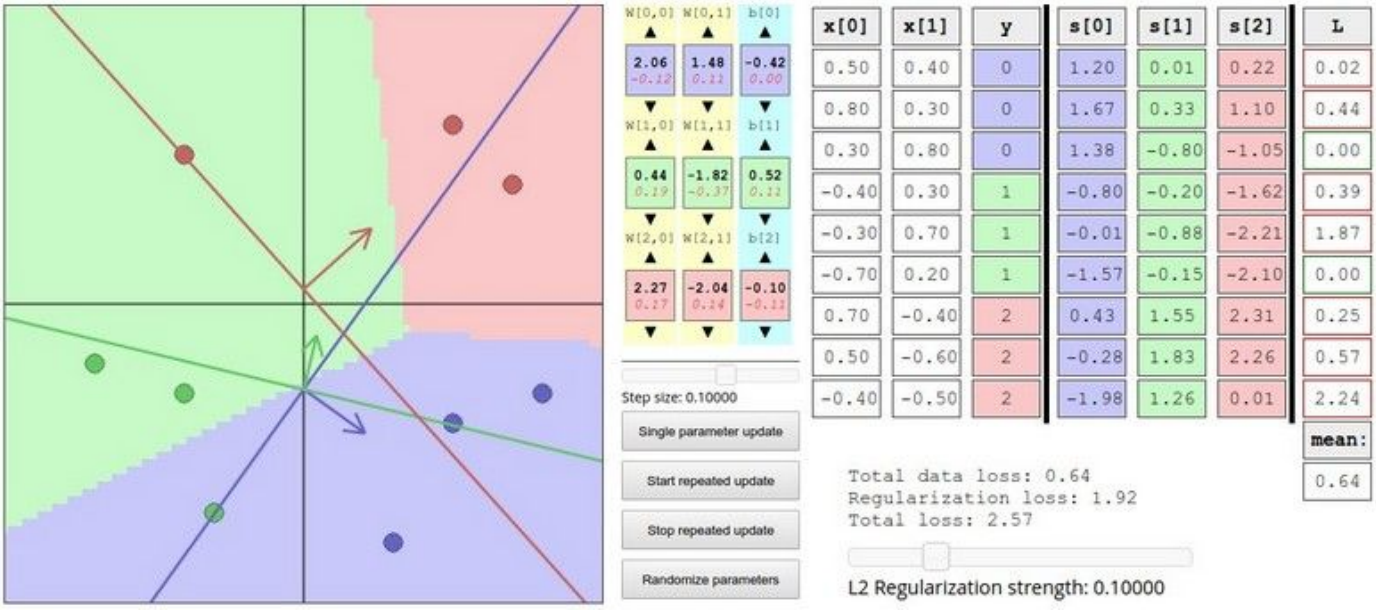
$$[0.5, -1, 0] \rightarrow [e^{0.5}, e^{-1}, e^0] = [1.65, 0.73, 1] \rightarrow [0.55, 0.12, 0.33]$$

现在看起来，概率的分布就更加分散了。还有，随着正则化参数λ不断增强，权重数值会越来越小，最后输出的概率会接近于均匀分布。这就是说，softmax分类器算出来的概率最好是看成一种对于分类正确性的自信。和SVM一样，数字间相互比较得出的大小顺序是可以解释的，但其绝对值则难以直观解释。

在实际使用中，SVM和Softmax经常是相似的：通常说来，两种分类器的表现差别很小，不同的人对于哪个分类器更好有不同的看法。相对于Softmax分类器，SVM更加“局部目标化（local objective）”，这既可以看做是一个特性，也可以看做是一个劣势。考虑一个评分是[10, -2, 3]的数据，其中第一个分类是正确的。那么一个SVM（$\Delta = 1$）会看到正确分类相较于不正确分类，已经得到了比边界值还要高的分数，它就会认为损失值是0。SVM对于数字个体的细节是不关心的：如果分数是[10, -100, -100]或者[10, 9, 9]，对于SVM来说没设么不同，只要满足超过边界值等于1，那么损失值就等于0。

对于softmax分类器，情况则不同。对于[10, 9, 9]来说，计算出的损失值就远远高于[10, -100, -100]的。换句话来说，softmax分类器对于分数是永远不会满意的：正确分类总能得到更高的可能性，错误分类总能得到更低的可能性，损失值总是能够更小。但是，SVM只要边界值被满足了就满意了，不会超过限制去细微地操作具体分数。这可以被看做是SVM的一种特性。举例说来，一个汽车的分类器应该把他的大量精力放在如何分辨小轿车和大卡车上，而不应该纠结于如何与青蛙进行区分，因为区分青蛙得到的评分已经足够低了。

交互式的网页Demo



我们实现了一个交互式的网页原型，来帮助读者直观地理解线性分类器。原型将损失函数进行可视化，画面表现的是对于2维数据的3种类别的分类。原型在课程进度上稍微超前，展现了最优化的内容，最优化将在下一节课讨论。

—

小结

总结如下：

定义了从图像像素映射到不同类别的分类评分的评分函数。在本节中，评分函数是一个基于权重**W**和偏差**b**的线性函数。
与kNN分类器不同，**参数方法**的优势在于一旦通过训练学习到了参数，就可以将训练数据丢弃了。同时该方法对于新的测试数据的预测非常快，因为只需要与权重**W**进行一个矩阵乘法运算。
介绍了偏差技巧，让我们能够将偏差向量和权重矩阵合二为一，然后就可以只跟踪一个矩阵。
定义了损失函数（介绍了SVM和Softmax线性分类器最常用的2个损失函数）。损失函数能够衡量给出的参数集与训练集数据真实类别情况之间的一致性。在损失函数的定义中可以看到，对训练集数据做出良好预测与得到一个足够低的损失值这两件事是等价的。

现在我们知道了如何基于参数，将数据集中的图像映射成为分类的评分，也知道了两种不同的损失函数，它们都能用来衡量算法分类预测的质量。但是，如何高效地得到能够使损失值最小的参数呢？这个求得最优参数的过程被称为最优化，将在下节课中进行介绍。