linear_classifier个人学习过程+理解

SVM损失和梯度：

对于每个样本 $X[i]$，其真实标签为 $y[i]$。我们计算它在所有类别上的得分 $scores = X[i]@W$。

对于所有**错误**的类别 $j != y[i]$，我们计算 $margin = scores[j] - scores[y[i]] + delta$ (代码中 $delta = 1$)。这个 $delta$ 常数是用于确定“正确的得分应该比错误的得分高出多少”的常数，实际操作中应该可以自己灵活调整（？）。

如果 margin > 0，也就是说明 $scores[y[i]]$ 比 $scores[j]$ 高，但高得**不够多**（没有达到 $delta$ 的差距），或者 $scores[j]$ 甚至比 $scores[y[i]]$ 还高，就代表了模型在这个错误类别 j 上“违反规定”了。那么则将其累加到总损失中。

最终损失是所有样本的平均损失加上正则化项 $reg * sum(W^2)$。

为什么需要正则化（Regularization）？

没有正则化可能导致过拟合（对于过拟合，我觉得【漫士】为什么刷题想得越多，考得反而越差？这个b站视频讲的很有启发性），如果没有正则化，机器只照着几张特定的训练数据拼命练习，把每一个细节，甚至数据上的一个小污点都模仿得一模一样。结果是：模型在训练数据上表现完美，但是，让其遇到一张新的、没见过的测试数据时，他可能就画得很糟糕，因为模型只学会了模仿那几张特定的训练数据，没有掌握普遍的规律。也就是，“过拟合”了。

梯度是什么?

在梯度下降 ($GradientDescent$) 中，我们想要减小损失。而梯度指向的是损失增加最快的方向。所以，我们沿着负梯度方向 ($-\nabla WL$) 更新权重 W，就能有效地降低损失。

在代码中，dW 就是损失函数的梯度，其本质上是一个多维倒数，表示如果稍微改变一点点 W 中的某个值。$Loss$ 会相应地改变多少，以及是增加还是减少。(dW 指向的是让 $Loss$ 增加最快的方向，所以实际调整的时候会需要负号)

所以权重 $W_{新} = W_{旧} - 学习率 * dW$

Softmax

和 SVM 的关系（区别）？

SUM 侧重于在给正确和错误的预测数据打一个明确的界限，尤其是在一些模糊的边界数据上进行处理。也就是说，尽量给一个评分标准使得正确的结果就是正确的，于错误的结果之间有明确的界限。

而 $Softmax$ 侧重于针对每一个可能的决策方向都给出一个“归属概率”，也就是给出所有可能性的概率，我们想要让其中最大的概率接近100%。

什么是 $Softmax$？

个人感觉其本质和 SVM 相近，对于不同类别都是线性计算得分。区别主要在于对于边界的处理:

$Softmax$ 的目标是将原始得分解释为概率。原始得分可以是任何实数（可能正、负、大、小）。为了将它们转化为满足以下条件的概率：

1. 非负（而且介于 0 和 1 之间）
2. 所有类别的概率之和为 1

引入了$Softmax$函数：

$$P(y = k|X_i) = p_k = exp(score_k)/\Sigma_j exp(score_j)$$

（个人感觉，这里用到指数函数，很精妙，他可以使得大的得分更加放大，同时把负数也掰正了。）所以对于每个输入 X_i，$Softmax$ 分类器输出一个覆盖所有 C个类别的概率分布。p_k 代表模型估计输入 X_i 属于类别 k 的概率。

小技巧，为了防止exp太大超过数据范围，一般会把所有数据减去最大的数据。

交叉熵损失?

$L_i = -\Sigma_k[\text{真实概率}(k) * log(\text{预测概率}(k))]$

带入数据也就是：

$L_i = -1 * log(\text{预测概率}(y_i)) = -log(p_{y_i})$

所以单个样本的损失就是模型赋予正确类别的概率的负对数值。

对于其他的（正则化&数据梯度的处理，与SVM没有本质区别）。