有时候，我们希望使用显式的限制，而不是惩罚。如第4.4节所描述，我们能修改下降算法（如随机梯度下降算法），使其先计算 $J(\boldsymbol{\theta})$ 的下降步，然后将 $\boldsymbol{\theta}$ 投影到满足 $\Omega(\boldsymbol{\theta}) < k$ 的最近点。如果我们知道什么样的 k 是合适的，而不想花时间寻找对应于此 k 处的 α 值，这会非常有用。

另一个使用显式约束和重投影而不是使用惩罚强加约束的原因是惩罚可能导致非凸优化过程而陷入局部极小（对应于小的 $\boldsymbol{\theta}$）。当训练神经网络时，这通常表现为训练带有几个“死亡单元”的神经网络。这些单元不会对网络学到的函数的行为有太大贡献，因为进入或离开他们的权重都非常小。当使用权重范数的惩罚训练时，即使可能通过增加权重以显著减少 J，这些配置也可能是局部最优的。因为重投影实现的显式约束不鼓励权重接近原点，所以在这些情况下工作得更好。通过重投影实现的显式约束只在权重变大并试图离开限制区域时产生作用。

最后，因为重投影的显式约束还对优化过程增加了一定的稳定性，所以这是另一个好处。当使用较高的学习率时，很可能进入正反馈，即大的权重诱导大梯度，然后诱发权重的大更新。如果这些更新持续增加权重的大小，$\boldsymbol{\theta}$ 就会迅速增大，直到离原点很远而发生溢出。重投影的显式约束可以防止这种反馈环引起的权重无限制的持续增加。Hinton *et al.* (2012b) 建议结合使用约束和高学习速率，这样能更快地探索参数空间，并保持一定的稳定性。

Hinton *et al.* (2012b) 尤其推荐由Srebro and Shraibman (2005) 引入的策略：约束神经网络层的权重矩阵每列的范数，而不是限制整个权重矩阵的Frobenius范数。分别限制每一列的范数可以防止某一隐藏单元有非常大的权重。如果我们将此约束转换成Lagrange函数中的一个惩罚，这将与 L^2 权重衰减类似但每个隐藏单元的权重都具有单独的KKT乘子。每个KKT乘子分别会被动态更新，以使每个隐藏单元服从约束。在实践中，列范数的限制总是通过重投影的显式约束实现。