```
import numpy,layers,sklearn,sklearn.metrics

# Monitor local slope of the decision function

def AvgGrad(nn,X): return ((nn.arch.backward(nn.arch.forward(X)*0+1)**2).sum
(axis=1)**.5).mean()
def MixNorm(nn): W = nn.arch.layers[0].W; return W.shape[1]**-.5 * (numpy.abs
(W).sum(axis=1)**2).sum()**.5

# Norm derivatives

class LinearFrob(layers.Linear):

    def computepgrad(self):
        DWErr = numpy.dot(self.A.T,self.DZ)/len(self.A)
        DWPen = 2*self.W
        self.DW = DWErr+0.001*DWPen
        self.DB = self.DZ.mean(axis=0)

class LinearMix(layers.Linear):

    def computepgrad(self):
        DWErr = numpy.dot(self.A.T,self.DZ)/len(self.A)
        DWPen = 2*numpy.abs(self.W).sum(axis=1)[:,numpy.newaxis]*numpy.sign
(self.W)/self.W.shape[1]
        self.DW = DWErr + 0.001*DWPen
        self.DB = self.DZ.mean(axis=0)
```