# Vanishing-Gradient

December 22, 2017

## 1 The Vanishing Gradient Problem

```
In [1]: import numpy as np
        import theano
        import theano.tensor as T
        import matplotlib.pyplot as plt
        from sklearn.utils import shuffle
        from util import relu, error_rate, getKaggleMNIST, init_weights
```

```
In [2]: class HiddenLayer(object):
            def __init__(self, D, M):
                W = init_weights((D, M))
                b = np.zeros(M)
                self.W = theano.shared(W)
                self.b = theano.shared(b)
                self.params = [self.W, self.b]

            def forward(self, X):
                return T.nnet.sigmoid(X.dot(self.W) + self.b)
```

```
In [7]: class ANN(object):
            def __init__(self, hidden_layer_sizes):
                self.hidden_layer_sizes = hidden_layer_sizes

            def fit(self, X, Y, learning_rate=0.01, mu=0.99, epochs=30, batch_sz=100):
                N, D = X.shape
                K = len(set(Y))

                self.hidden_layers = []
                mi = D
                for mo in self.hidden_layer_sizes:
                    h = HiddenLayer(mi, mo)
                    self.hidden_layers.append(h)
                    mi = mo

                # initialize logistic regression layer
                W = init_weights((mo, K))
```

```
b = np.zeros(K)
self.W = theano.shared(W)
self.b = theano.shared(b)

self.params = [self.W, self.b]
self.allWs = []
for h in self.hidden_layers:
    self.params += h.params
    self.allWs.append(h.W)
self.allWs.append(self.W)

X_in = T.matrix('X_in')
targets = T.ivector('Targets')
pY = self.forward(X_in)

cost = -T.mean( T.log(pY[T.arange(pY.shape[0]), targets]) )
prediction = self.predict(X_in)
# cost_predict_op = theano.function(
#     inputs=[X_in, targets],
#     outputs=[cost, prediction],
# )

dparams = [theano.shared(p.get_value()*0) for p in self.params]
grads = T.grad(cost, self.params)

updates = [
    (p, p + mu*dp - learning_rate*g) for p, dp, g in zip(self.params, dparams, g
] + [
    (dp, mu*dp - learning_rate*g) for dp, g in zip(dparams, grads)
]
train_op = theano.function(
    inputs=[X_in, targets],
    outputs=[cost, prediction],
    updates=updates,
)

n_batches = N / batch_sz
costs = []
lastWs = [W.get_value() for W in self.allWs]
W_changes = []
print ("supervised training...")
for i in range(int(epochs)):
    print ("epoch:", i)
    X, Y = shuffle(X, Y)
    for j in range(int(n_batches)):
        Xbatch = X[j*batch_sz:(j*batch_sz + batch_sz)]
        Ybatch = Y[j*batch_sz:(j*batch_sz + batch_sz)]
        c, p = train_op(Xbatch, Ybatch)
```

```
                if j % 100 == 0:
                    print ("j / n_batches:", j, "/", n_batches, "cost:", c, "error:", er
                costs.append(c)

                # log changes in all Ws
                W_change = [np.abs(W.get_value() - lastW).mean() for W, lastW in zip(sel
                W_changes.append(W_change)
                lastWs = [W.get_value() for W in self.allWs]

        W_changes = np.array(W_changes)
        plt.subplot(2,1,1)
        for i in range(W_changes.shape[1]):
            plt.plot(W_changes[:,i], label='layer %s' % i)
        plt.legend()
        # plt.show()

        plt.subplot(2,1,2)
        plt.plot(costs)
        plt.show()

    def predict(self, X):
        return T.argmax(self.forward(X), axis=1)

    def forward(self, X):
        Z = X
        for h in self.hidden_layers:
            Z = h.forward(Z)
        Y = T.nnet.softmax(Z.dot(self.W) + self.b)
        return Y
```

In [8]:
```
Xtrain, Ytrain, Xtest, Ytest = getKaggleMNIST()
dnn = ANN([1000, 750, 500])
dnn.fit(Xtrain, Ytrain)
```

```
supervised training...
epoch: 0
j / n_batches: 0 / 410.0 cost: 2.341391721004479 error: 0.9
j / n_batches: 100 / 410.0 cost: 2.3574872676083345 error: 0.9
j / n_batches: 200 / 410.0 cost: 2.301575629924423 error: 0.89
j / n_batches: 300 / 410.0 cost: 2.298286594655491 error: 0.88
j / n_batches: 400 / 410.0 cost: 2.298375732101419 error: 0.91
epoch: 1
j / n_batches: 0 / 410.0 cost: 2.3043720629054016 error: 0.86
j / n_batches: 100 / 410.0 cost: 2.30891806720224 error: 0.88
j / n_batches: 200 / 410.0 cost: 2.2961686467011377 error: 0.82
j / n_batches: 300 / 410.0 cost: 2.3168475795196217 error: 0.94
j / n_batches: 400 / 410.0 cost: 2.3029717594495893 error: 0.88
epoch: 2
```

```
j / n_batches: 0 / 410.0 cost: 2.3272877975258144 error: 0.89
j / n_batches: 100 / 410.0 cost: 2.2953590825470847 error: 0.84
j / n_batches: 200 / 410.0 cost: 2.3156640007759983 error: 0.9
j / n_batches: 300 / 410.0 cost: 2.312399534099112 error: 0.91
j / n_batches: 400 / 410.0 cost: 2.2980735778131662 error: 0.84
epoch: 3
j / n_batches: 0 / 410.0 cost: 2.3193381877787793 error: 0.93
j / n_batches: 100 / 410.0 cost: 2.3004956406890416 error: 0.89
j / n_batches: 200 / 410.0 cost: 2.3188476539139433 error: 0.94
j / n_batches: 300 / 410.0 cost: 2.3038853867873743 error: 0.86
j / n_batches: 400 / 410.0 cost: 2.2934473431663025 error: 0.86
epoch: 4
j / n_batches: 0 / 410.0 cost: 2.30092746092822 error: 0.92
j / n_batches: 100 / 410.0 cost: 2.302987817448466 error: 0.92
j / n_batches: 200 / 410.0 cost: 2.2916152955102786 error: 0.92
j / n_batches: 300 / 410.0 cost: 2.317390125596637 error: 0.92
j / n_batches: 400 / 410.0 cost: 2.3145308944013907 error: 0.93
epoch: 5
j / n_batches: 0 / 410.0 cost: 2.3002980186715134 error: 0.88
j / n_batches: 100 / 410.0 cost: 2.3194436162157674 error: 0.89
j / n_batches: 200 / 410.0 cost: 2.308650228905738 error: 0.88
j / n_batches: 300 / 410.0 cost: 2.2996358231981278 error: 0.88
j / n_batches: 400 / 410.0 cost: 2.2966614021796015 error: 0.86
epoch: 6
j / n_batches: 0 / 410.0 cost: 2.3091874340568386 error: 0.89
j / n_batches: 100 / 410.0 cost: 2.299024445209212 error: 0.95
j / n_batches: 200 / 410.0 cost: 2.313819490539815 error: 0.92
j / n_batches: 300 / 410.0 cost: 2.292295714136435 error: 0.89
j / n_batches: 400 / 410.0 cost: 2.30971364976027 error: 0.91
epoch: 7
j / n_batches: 0 / 410.0 cost: 2.306951204416041 error: 0.92
j / n_batches: 100 / 410.0 cost: 2.3025778180837717 error: 0.88
j / n_batches: 200 / 410.0 cost: 2.3174088512247537 error: 0.91
j / n_batches: 300 / 410.0 cost: 2.303958834595984 error: 0.91
j / n_batches: 400 / 410.0 cost: 2.2903614864263613 error: 0.87
epoch: 8
j / n_batches: 0 / 410.0 cost: 2.308807382163972 error: 0.89
j / n_batches: 100 / 410.0 cost: 2.295052897553628 error: 0.85
j / n_batches: 200 / 410.0 cost: 2.3102741608154935 error: 0.9
j / n_batches: 300 / 410.0 cost: 2.3008136454687667 error: 0.92
j / n_batches: 400 / 410.0 cost: 2.2995390207975803 error: 0.88
epoch: 9
j / n_batches: 0 / 410.0 cost: 2.3064524795667496 error: 0.84
j / n_batches: 100 / 410.0 cost: 2.3035138605536805 error: 0.91
j / n_batches: 200 / 410.0 cost: 2.3141217984733147 error: 0.92
j / n_batches: 300 / 410.0 cost: 2.3244005614872547 error: 0.94
j / n_batches: 400 / 410.0 cost: 2.3057480267449275 error: 0.91
epoch: 10
```

```
j / n_batches: 0 / 410.0 cost: 2.3080647067054114 error: 0.91
j / n_batches: 100 / 410.0 cost: 2.288823096055822 error: 0.87
j / n_batches: 200 / 410.0 cost: 2.311299049794977 error: 0.91
j / n_batches: 300 / 410.0 cost: 2.307376293112833 error: 0.94
j / n_batches: 400 / 410.0 cost: 2.2999118408783703 error: 0.91
epoch: 11
j / n_batches: 0 / 410.0 cost: 2.320508120587295 error: 0.87
j / n_batches: 100 / 410.0 cost: 2.312447337657224 error: 0.94
j / n_batches: 200 / 410.0 cost: 2.3078666614803947 error: 0.93
j / n_batches: 300 / 410.0 cost: 2.3029991409755786 error: 0.87
j / n_batches: 400 / 410.0 cost: 2.3186921912421785 error: 0.91
epoch: 12
j / n_batches: 0 / 410.0 cost: 2.3001273604512615 error: 0.87
j / n_batches: 100 / 410.0 cost: 2.3098441981664357 error: 0.94
j / n_batches: 200 / 410.0 cost: 2.3113827933746123 error: 0.93
j / n_batches: 300 / 410.0 cost: 2.2985946237841604 error: 0.91
j / n_batches: 400 / 410.0 cost: 2.297873991228372 error: 0.86
epoch: 13
j / n_batches: 0 / 410.0 cost: 2.3020240266939656 error: 0.91
j / n_batches: 100 / 410.0 cost: 2.3105800313013636 error: 0.86
j / n_batches: 200 / 410.0 cost: 2.3049964678736843 error: 0.89
j / n_batches: 300 / 410.0 cost: 2.299802205098634 error: 0.86
j / n_batches: 400 / 410.0 cost: 2.2963605236986586 error: 0.85
epoch: 14
j / n_batches: 0 / 410.0 cost: 2.3004622246607362 error: 0.86
j / n_batches: 100 / 410.0 cost: 2.2987275732984256 error: 0.87
j / n_batches: 200 / 410.0 cost: 2.2955070287040993 error: 0.83
j / n_batches: 300 / 410.0 cost: 2.299182125989684 error: 0.94
j / n_batches: 400 / 410.0 cost: 2.3045512327110567 error: 0.88
epoch: 15
j / n_batches: 0 / 410.0 cost: 2.313146959033381 error: 0.94
j / n_batches: 100 / 410.0 cost: 2.2836413912553537 error: 0.7
j / n_batches: 200 / 410.0 cost: 2.1528489841064467 error: 0.74
j / n_batches: 300 / 410.0 cost: 2.07544663543642 error: 0.85
j / n_batches: 400 / 410.0 cost: 2.113444419418048 error: 0.73
epoch: 16
j / n_batches: 0 / 410.0 cost: 1.825649239194199 error: 0.65
j / n_batches: 100 / 410.0 cost: 1.748876847840169 error: 0.6
j / n_batches: 200 / 410.0 cost: 1.735124137004177 error: 0.7
j / n_batches: 300 / 410.0 cost: 1.6601686128120405 error: 0.61
j / n_batches: 400 / 410.0 cost: 1.673591056553175 error: 0.69
epoch: 17
j / n_batches: 0 / 410.0 cost: 1.6879114983798693 error: 0.66
j / n_batches: 100 / 410.0 cost: 1.7156600772988175 error: 0.7
j / n_batches: 200 / 410.0 cost: 1.584177158217764 error: 0.69
j / n_batches: 300 / 410.0 cost: 1.5581721813542688 error: 0.57
j / n_batches: 400 / 410.0 cost: 1.4544278269223971 error: 0.62
epoch: 18
```

```
j / n_batches: 0 / 410.0 cost: 1.387380567098259 error: 0.62
j / n_batches: 100 / 410.0 cost: 1.3522515846856222 error: 0.52
j / n_batches: 200 / 410.0 cost: 1.2471036255078682 error: 0.52
j / n_batches: 300 / 410.0 cost: 1.0224684225249367 error: 0.38
j / n_batches: 400 / 410.0 cost: 0.7613122682911977 error: 0.24
epoch: 19
j / n_batches: 0 / 410.0 cost: 0.7874021562379337 error: 0.24
j / n_batches: 100 / 410.0 cost: 0.6537619911145378 error: 0.2
j / n_batches: 200 / 410.0 cost: 0.4993899783156091 error: 0.18
j / n_batches: 300 / 410.0 cost: 0.5402028562410719 error: 0.19
j / n_batches: 400 / 410.0 cost: 0.39393118818810513 error: 0.12
epoch: 20
j / n_batches: 0 / 410.0 cost: 0.7487238063637325 error: 0.23
j / n_batches: 100 / 410.0 cost: 0.29517124728278565 error: 0.06
j / n_batches: 200 / 410.0 cost: 0.5237015496507073 error: 0.19
j / n_batches: 300 / 410.0 cost: 0.5357526264361937 error: 0.18
j / n_batches: 400 / 410.0 cost: 0.2909770497539497 error: 0.12
epoch: 21
j / n_batches: 0 / 410.0 cost: 0.24073158441523138 error: 0.08
j / n_batches: 100 / 410.0 cost: 0.2919206854560198 error: 0.11
j / n_batches: 200 / 410.0 cost: 0.44691764824525615 error: 0.11
j / n_batches: 300 / 410.0 cost: 0.2874699182244036 error: 0.06
j / n_batches: 400 / 410.0 cost: 0.4064966725800365 error: 0.12
epoch: 22
j / n_batches: 0 / 410.0 cost: 0.24661979119974464 error: 0.11
j / n_batches: 100 / 410.0 cost: 0.24468659605939075 error: 0.07
j / n_batches: 200 / 410.0 cost: 0.24332169680972981 error: 0.09
j / n_batches: 300 / 410.0 cost: 0.32278502435284784 error: 0.1
j / n_batches: 400 / 410.0 cost: 0.2492759710220849 error: 0.06
epoch: 23
j / n_batches: 0 / 410.0 cost: 0.2396932884410754 error: 0.06
j / n_batches: 100 / 410.0 cost: 0.2013135920601152 error: 0.03
j / n_batches: 200 / 410.0 cost: 0.15106811049586297 error: 0.04
j / n_batches: 300 / 410.0 cost: 0.18349342813370573 error: 0.05
j / n_batches: 400 / 410.0 cost: 0.2415165505048341 error: 0.08
epoch: 24
j / n_batches: 0 / 410.0 cost: 0.0987911251028836 error: 0.03
j / n_batches: 100 / 410.0 cost: 0.1975644943504079 error: 0.07
j / n_batches: 200 / 410.0 cost: 0.26727849056819325 error: 0.12
j / n_batches: 300 / 410.0 cost: 0.18499113768714956 error: 0.04
j / n_batches: 400 / 410.0 cost: 0.24045673820531654 error: 0.08
epoch: 25
j / n_batches: 0 / 410.0 cost: 0.18547729381624614 error: 0.05
j / n_batches: 100 / 410.0 cost: 0.10357067308887485 error: 0.04
j / n_batches: 200 / 410.0 cost: 0.06358817953662803 error: 0.02
j / n_batches: 300 / 410.0 cost: 0.22502699958928105 error: 0.08
j / n_batches: 400 / 410.0 cost: 0.1369264898223722 error: 0.07
epoch: 26
```

```
j / n_batches: 0 / 410.0 cost: 0.23538909668186914 error: 0.07
j / n_batches: 100 / 410.0 cost: 0.1583717444052073 error: 0.05
j / n_batches: 200 / 410.0 cost: 0.24525211189784277 error: 0.06
j / n_batches: 300 / 410.0 cost: 0.09288194554895161 error: 0.03
j / n_batches: 400 / 410.0 cost: 0.07509880221381332 error: 0.02
epoch: 27
j / n_batches: 0 / 410.0 cost: 0.2448222235856519 error: 0.05
j / n_batches: 100 / 410.0 cost: 0.07872984728990744 error: 0.02
j / n_batches: 200 / 410.0 cost: 0.22061674050623317 error: 0.06
j / n_batches: 300 / 410.0 cost: 0.2747103537269592 error: 0.07
j / n_batches: 400 / 410.0 cost: 0.16438690411742854 error: 0.04
epoch: 28
j / n_batches: 0 / 410.0 cost: 0.18958626964371578 error: 0.04
j / n_batches: 100 / 410.0 cost: 0.11593695460446073 error: 0.02
j / n_batches: 200 / 410.0 cost: 0.061713432260489504 error: 0.02
j / n_batches: 300 / 410.0 cost: 0.14235569557898395 error: 0.05
j / n_batches: 400 / 410.0 cost: 0.17173646562066555 error: 0.06
epoch: 29
j / n_batches: 0 / 410.0 cost: 0.16818241553984803 error: 0.04
j / n_batches: 100 / 410.0 cost: 0.10568032498721033 error: 0.03
j / n_batches: 200 / 410.0 cost: 0.17065047569605019 error: 0.06
j / n_batches: 300 / 410.0 cost: 0.16179063311056516 error: 0.05
j / n_batches: 400 / 410.0 cost: 0.09003309893868978 error: 0.03
```