Homework 7: Due Tue 10-02-2018
Total Points (20 pts)

Stochastic Gradient Descent

Use stochastic gradient descent to train a logistic regression classifier for the Iris dataset `Iris-cleaned.csv`. Use `Dloss_numeric` to check the accuracy of `Dloss`. Compare your results to results obtained using Keras. You may find the code provided below helpful.

```
import numpy as np
import pandas as pd
from keras.models import Sequential
from keras.layers.core import Dense, Activation
from keras.optimizers import Adam

def fsmax(x):
    e = np.exp(x-x.max())
    p = e/e.sum()
    return p

def Dfsmax(x):
    e = np.exp(x-x.max())
    p = e/e.sum()
    dpdx = np.diag(p) - np.outer(p,p)
    return dpdx

def CE(p,ph):
    ce = <...code ommitted...>
    return ce

def loss(x,p,W):

    <...code ommitted...>

    return l

def TotalLoss(X,P,W):

    <...code ommitted...>

    return l

def Dloss(x,p,W):
    yh = np.matmul(x,W)
    ph = fsmax(yh)
    dpdy = Dfsmax(yh)
    u = p/ph
    v = np.matmul(u,dpdy)
    dldW = - np.outer(x,v)
    return dldW
```

(continued on next page)

```
        def Dloss_numeric(x,p,W):
            eps = 10**(-5)
            (d,N) = W.shape
            d = d-1
            dldW = np.zeros((d+1,N))
            for j in range(d+1):
                for k in range(N):
                    dW = np.zeros((d+1,N))
                    dW[j,k] = eps
                    l_p = loss(x,p,W+dW)
                    l_m = loss(x,p,W-dW)
                    dldW[j,k] = (l_p - l_m)/(2*eps)
            return dldW

        def accuracy(X,P,W):
            n = X.shape[0]
            Yh = np.matmul(X,W)
            Ph = np.zeros(Yh.shape)
            for i in range(n):
                Ph[i,:] = fsmax(Yh[i,:])
            num_correct = (Ph.argmax(axis=1) == P.argmax(axis=1)).sum()
            acc = num_correct/n
            return acc

        def StochasticGradientDescent(X,P,W,h,epochs,verbose=0):

            <...code ommitted...>

            if verbose!=0:
              print('epoch =',epoch+1,'of',epochs,' loss =',l,' acc =',np.round(acc,3))
            return (W,l,acc)

        # use_bias=False to allow user to provide a bias column instead of Keras
            model.add(Dense(3,input_shape=(5,),use_bias=False))
```