Нейронные сети, классификация 1 слой

С.И.Хашин
http://math.ivanovo.ac.ru/dalgebra/Khashin/index.html

Ивановский государственный университет

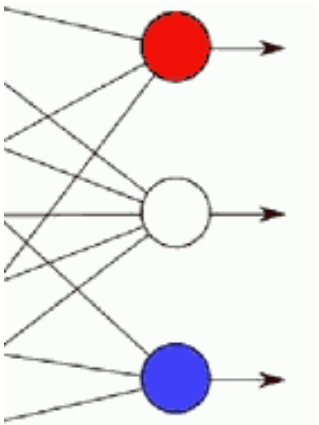
.

Иваново-2019

План

SoftMax

Мы решаем задачу распознавания объектов на 3 класса (кошка/собака/ни то ни другое). Тогда мы строим нейронную сеть, имеющую 3 выходных нейрона:



причем с тривиальной функцией активации (т.е. $f(t) = t$).
Выходные значения этих 3-х нейронов обозначим (v_0, v_1, v_2).
Если v_0 максимальное среди них, то ответ — 0.
Если v_1 максимальное среди них, то ответ — 1.
Если v_2 максимальное среди них, то ответ — 2.

SoftMax

В процессе обучения нейросети в каждый момент времени у каждого нейрона есть свой набор внутренних параметров.
Имеются обучающие вектора X_i и верный ответ Y_i на каждом из этих векторов.
При задаче регрессии имеем только один выходной нейрон, его значение на векторе X_i обозначим Y_i'. Целевой функцией в этом случаем будет сумма квадратов отклонений:

$$S = \sum (Y_i' - Y_i)^2.$$

А какую целевую функцию брать для задачи классификации?
Предположим, что на некотором входном векторе нейросеть выдала 3 значения: $(v_0, v_1, v_2) = (-1, 3, 2)$, при этом верным ответом на векторе X_i является 2.

SoftMax

Итак: $(v_0, v_1, v_2) = (-1, 3, 2)$, при этом верным ответом на входном векторе является 2.
Сначала найдем числа z_i:

$$z_0 = \frac{e^{v_0}}{e^{v_0} + e^{v_1} + e^{v_2}}, \quad z_1 = \frac{e^{v_1}}{e^{v_0} + e^{v_1} + e^{v_2}}, \quad z_2 = \frac{e^{v_2}}{e^{v_0} + e^{v_1} + e^{v_2}}.$$

Видим, что $z_i > 0$ и $z_0 + z_1 + z_2 = 1$.
Если на входном векторе верным ответом является $k \in \{0, 1, 2\}$, то к целевой функции прибавим $-\ln z_k$.

SoftMax

При $(v_0, v_1, v_2) = (-1, 3, 2)$,
$e^{v_0} + e^{v_1} + e^{v_2} \approx 0.37 + 20.1 + 7.4 \approx 27.8$ и
$z_i = (0.013, 0.721, 0.266)$.
Так как $k = 2$, то вклад в целевую функцию будет равен
$S_i = -\ln 0.266 \approx 1.33$. То есть

$$S_i = -\ln \frac{e^{v_2}}{e^{v_0} + e^{v_1} + e^{v_2}} = -(\ln e^{v_2} - \ln(e^{v_0} + e^{v_1} + e^{v_2}))$$

$$= \ln(e^{v_0} + e^{v_1} + e^{v_2}) - v_2 > 0.$$

Задание. Написать формулу на Excel и проверить, какие значения она будет принимать при разных (v_0, v_1, v_2).

SoftMax градиент

Если верным ответом на входном векторе является 2, то

$$S_i = \ln(e^{v_0} + e^{v_1} + e^{v_2}) - v_2$$

и

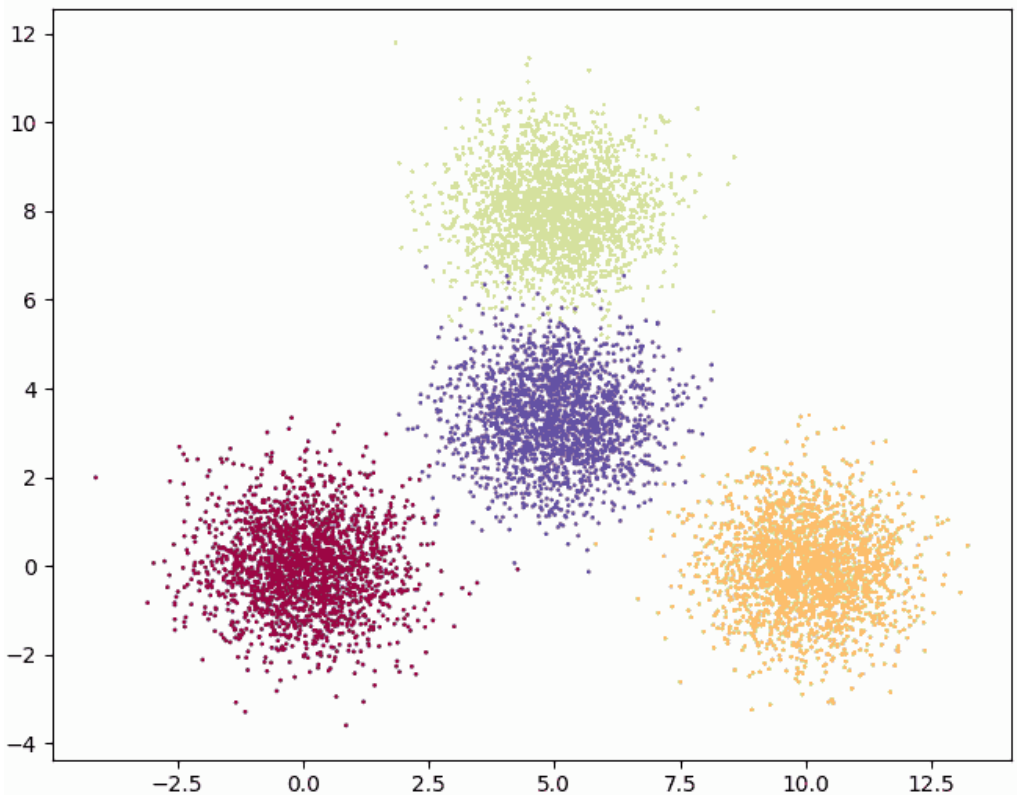
$$\frac{\partial S_i}{\partial v_0} = \frac{e^{v_0}}{e^{v_0} + e^{v_1} + e^{v_2}} > 0,$$

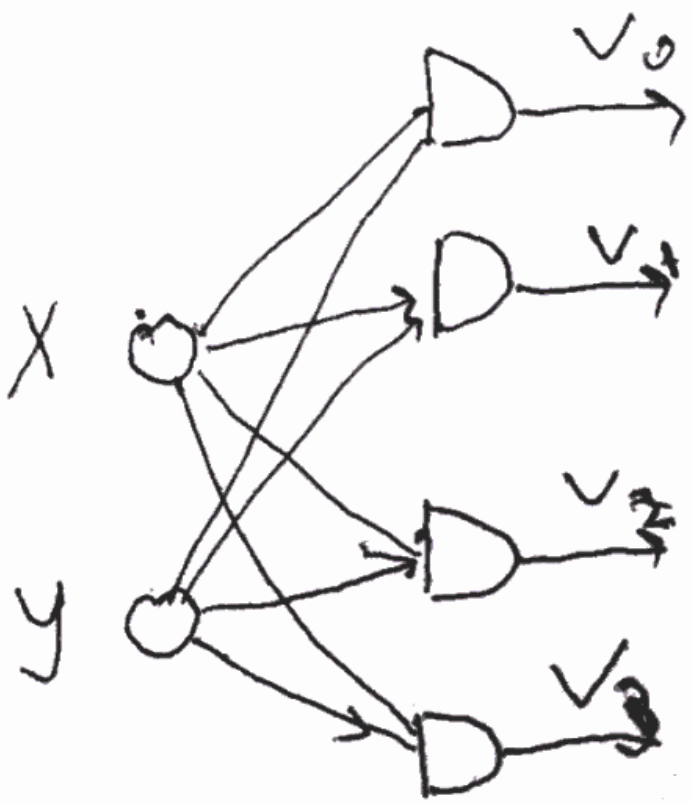
$$\frac{\partial S_i}{\partial v_1} = \frac{e^{v_1}}{e^{v_0} + e^{v_1} + e^{v_2}} > 0,$$

$$\frac{\partial S_i}{\partial v_2} = \frac{e^{v_2}}{e^{v_0} + e^{v_1} + e^{v_2}} - 1 < 0,$$

4 круга, 4 нейрона, однослойная сеть

На входе (x, y), на выходе (v_0, v_1, v_2, v_3):





4 круга, 4 нейрона, однослойная сеть

$(x, y) \to (v_0, v_1, v_2, v_3)$:

$$\begin{aligned} v_0 &= w_{00} + w_{01}x + w_{02}x \\ v_1 &= w_{10} + w_{11}x + w_{12}x \\ v_2 &= w_{20} + w_{21}x + w_{22}x \\ v_3 &= w_{30} + w_{31}x + w_{32}x \end{aligned}$$

Если на векторе (x, y) верным ответом является 2, то значение целевой функции:

$$S_i = \ln(e^{v_0} + e^{v_1} + e^{v_2} + e^{v_3}) - v_2.$$

Если $v_2 >> (v_0, v_1, v_3)$, то $S_i \approx 0$.
Если все v_i примерно одинаковы, то $S_i \approx \ln 4 \approx 1.4$.

Однослойная нейронная сеть

Пусть X — обучающая матрица размера $N \times K$ и Y — вектор ответов длиной N, состоящий из целых числе от 0 до $M-1$, то есть задача классификации на M классов.
Берем M нейронов, каждый имеет K входов, то есть $K+1$ внутренний параметр:

$$(x_0, \ldots, x_{K-1}) \to w_0 + w_1 \cdot x_0 + \cdots + w_K \cdot x_{K-1}.$$

Параметры всех нейронов объединяем в одну матрицу

$$W = \begin{pmatrix} w_{00} & w_{01} & \ldots & w_{0K} \\ w_{10} & w_{11} & \ldots & w_{1K} \\ & & \ldots & \end{pmatrix}$$

Однослойная нейронная сеть

Если мы добавим к матрице X в начале столбец из единиц, то значения всех нейронов V можно получить умножением матрицы X на транспонированную к W:

```
V = X.dot(W.T) # shape = (N,M)
```

Теперь целевая функция на векторе v при верном ответе m (в нашем примере $v = (-1, 3, 2)$, $m = 2$:

```
def target1(v,m):
  return np.log(np.exp(v).sum()) - v[m]
```

SoftMax градиент

Пусть входные вектора имеют размерность $K = 2$ и число классов $M = 3$. Тогда

$$W = \begin{pmatrix} w_{00} & w_{01} & w_{02} \\ w_{10} & w_{11} & w_{12} \\ w_{20} & w_{21} & w_{22} \end{pmatrix}$$

и для входного вектора (x_0, x_1) получим:

$$\begin{aligned} v_0 &= w_{00} + w_{01} \cdot x_0 + w_{02} \cdot x_1 \\ v_1 &= w_{10} + w_{11} \cdot x_0 + w_{12} \cdot x_1 \\ v_2 &= w_{20} + w_{21} \cdot x_0 + w_{22} \cdot x_1 \end{aligned}$$

SoftMax градиент

Для входного вектора (x_0, x_1):

$$\begin{aligned} v_0 &= w_{00} + w_{01} \cdot x_0 + w_{02} \cdot x_1 \\ v_1 &= w_{10} + w_{11} \cdot x_0 + w_{12} \cdot x_1 \\ v_2 &= w_{20} + w_{21} \cdot x_0 + w_{22} \cdot x_1 \end{aligned}$$

Если верным ответом на входном векторе является 2, то

$$S_i = \ln(e^{v_0} + e^{v_1} + e^{v_2}) - v_2.$$

и

$$\frac{\partial S_i}{\partial v_0} = \frac{e^{v_0}}{e^{v_0} + e^{v_1} + e^{v_2}} > 0,$$

$$\frac{\partial S_i}{\partial v_1} = \frac{e^{v_1}}{e^{v_0} + e^{v_1} + e^{v_2}} > 0,$$

$$\frac{\partial S_i}{\partial v_2} = \frac{e^{v_2}}{e^{v_0} + e^{v_1} + e^{v_2}} - 1 < 0,$$

SoftMax градиент

$$\begin{aligned} v_0 &= w_{00} + w_{01} \cdot x_0 + w_{02} \cdot x_1 \\ v_1 &= w_{10} + w_{11} \cdot x_0 + w_{12} \cdot x_1 \\ v_2 &= w_{20} + w_{21} \cdot x_0 + w_{22} \cdot x_1 \end{aligned}$$

$$S_i = \ln\left(e^{v_0} + e^{v_1} + e^{v_2}\right) - v_2$$

Обозначим

$$z_i = \frac{e^{v_i}}{e^{v_0} + e^{v_1} + e^{v_2}}.$$

Тогда

$$\frac{\partial S_i}{\partial w_{00}} = \frac{\partial S_i}{\partial v_0} \cdot \frac{\partial v_0}{\partial w_{00}} = z_0.$$

$$\frac{\partial S_i}{\partial w_{01}} = \frac{\partial S_i}{\partial v_0} \cdot \frac{\partial v_0}{\partial w_{01}} = z_0 \cdot x_0.$$

$$\frac{\partial S_i}{\partial w_{02}} = \frac{\partial S_i}{\partial v_0} \cdot \frac{\partial v_0}{\partial w_{02}} = z_0 \cdot x_1.$$

Загрузка данных

```
def data_load(idx): ''' Загрузка данных
  :param idx: 0:4circles.csv,     1:spiral_02_03.csv,
              2:spiral_02_05.csv, 3:sky_data.csv,
              4:mnist_std.npz, 5:cifar10_std.npz
  :return: Y,X  '''
  if idx==0:
    data=np.loadtxt('4circles.csv',skiprows=1,delimiter=',')
    return data[:,0].astype(np.uint8), data[:,1:]
  if idx==1:
    data=np.loadtxt('spiral_02_03.csv',skiprows=1,delimiter=
    return data[:,0].astype(np.uint8), data[:,1:]
  if idx==2:
    data=np.loadtxt('spiral_02_05.csv',skiprows=1,delimiter=
    return data[:,0].astype(np.uint8), data[:,1:]
  ...
```

Загрузка данных

```
def data_load(idx): ''' Загрузка данных '''
  ...
  if idx==3:
    Ys = np.genfromtxt('sky_data.csv', skip_header=1,
              delimiter=',', usecols=13, dtype='str')
    # 0: STAR, 1:GALAXY, 2:QSO
    Y = np.zeros(len(Ys), dtype=np.uint8)
    Y[Ys=='GALAXY'] = 1
    Y[Ys=='QSO'] = 2
    # [ra, dec, u, g, r, i, z, run, camcol, field]
    X = np.genfromtxt('sky_data.csv', skip_header=1,
       delimiter=',', usecols=(1,2,3,4,5,6,7,8,10,11) )
    return Y,X
  if idx==4:
    data = np.load(r"D:\a\ML_datasets\!std\mnist_std.npz")
    return data['Y'], data['X'].astype(float)
```

Загрузка данных

```
def data_load(idx): ''' Загрузка данных '''
  ...
  if idx==5:
    data = np.load(r"D:\a\ML_datasets\!std\CIFAR10_std.npz")
    return data['Y'], data['X'].astype(float)

def tst_data_load():
  for i in range(6):
    Y,X = data_load(i)
    print(f'{i:2}, {Y.shape}, {X.shape}')
    print('   Y:', Y[:10])
    print('   X:', X[:4])
    print('----------------------')
```

Целевая функция

```
def target(Y, X, W):  # целевая функция
    V = X.dot(W.T)    # shape = (N,M)
    S = 0
    for i,v in enumerate(V):
        S +=  np.log(np.exp(v).sum()) - v[Y[i]]
    return S/len(Y)
```

Градиент для одной строки

```
def dTarget1(y1, x1, W):
    ''' Целевая функция и её градиент для одного
        обучающего вектора x1 и верного ответа y1
    :param y1: верный ответ, число [0,M)
    :param x1: обучающий вектор длины K+1
    :param W:  параметры нейронов, матрица размера (M, K+1)
    :return:(целевая функция, её градиент по всем w[i,j] in W
    '''
    v = x1.dot(W.T)  # вектор длины M
    z = np.exp(v)
    S0 = np.log(z.sum())-v[y1] #целевая функция на векторе x1
    z /= z.sum()
    z[y1] -= 1
    res = np.outer(z,x1)
    return S0, res
```

Градиент для всей матрицы

```
def dTarget(Y, X, W):
    ''' Целевая функция и её градиент
        для обучающей матрицы X и вектора ответов Y
    :param Y: вектор верных ответов длины N, числа [0,M)
    :param X: обучающая матрица (N,K+1)
    :param W:  параметры нейронов, матрица размера (M, K+1)
    :return:(целевая функция,её градиент по всем w[i,j] in W
    '''
    V = X.dot(W.T)  # shape = (N,M)
    S0 = 0; dS0 = np.zeros(W.shape)
    for i, x1 in enumerate(X):
        tt = dTarget1(Y[i], x1, W)
        S0 += tt[0]  # целевая функция
        dS0 += tt[1] # её градиент

    return S0/len(Y), dS0/len(Y)
```

Градиентый спуск

```
def gradient_descent(f, df, w0, LR, n_step=100):
  eps = 1e-10  # минимальное улучшение за один шаг
  w0 = np.array(w0)
  S0 = S1 = f(w0)
  iStep=-1
  for iStep in range(n_step):
    w1 = w0 - LR*np.array(df(w0))
    if np.linalg.norm(w1)>500: break
    S1 = f(w1)
    if S1 > S0-eps: break # слишком малое улучшение
    #if iStep%100==0: print(f'{iStep:4d}, {w1[:6]},  {S1:13
    w0, S0 = w1, S1
  return w0, S0, iStep
```

Целевая функция и градиент

Нам нужны функции в виде $f(w)$, $df(w)$:

```
X = Y = None
def f(w):   # целевая фунция, w - вектор
  global X, Y
  N, K = X.shape
  WW = w.reshape((-1,K))
  return target(Y, X, WW)

def df(w):   # градиент целевой фунции, w - вектор
  global X, Y
  N, K = X.shape
  WW = w.reshape((-1,K))
  return dTarget(Y, X, WW)[1].reshape(-1)
```

Результаты работы

```
def neuro_result(X,W): # результат работы нейросети с
                       # матрицей W на обучающих данных X
  return np.argmax(X.dot(W.T), axis=1)

def accuracy(Y,X,W):  # точность предсказания
  Ys = neuro_result(X,W)
  return np.count_nonzero(Y==Ys)/len(Y)
```

Вывод картинки на экран

```
def draw_dim2(Y,X,W, acc=0, idx=None):
    Ys = neuro_result(X,W)
    CX = X[:, 1]
    CY = X[:, 2]
    cx0 = CX[Ys==Y]
    cy0 = CY[Ys==Y]
    plt.scatter(cx0, cy0, s=1, color='blue' )
    cx1 = CX[Ys!=Y]
    cy1 = CY[Ys!=Y]
    plt.scatter(cx1, cy1, s=1, color='red' )
    plt.title(f'step {idx:4d} acc {acc:8.6f}')
    plt.show()
```

Run

```
def tst_01():   # общая проверка
  global Y,X
  #  0:4circles.csv, 1:spiral_02_03.csv, 2:spiral_02_05.csv
  #  3:sky_data.csv, 4:mnist_std.npz, 5:cifar10_std.npz
  Y,X = data_load(0)

  # нормализуем столбцы матрицы X
  X -= X.mean(axis=0) # Из каждого столбца матрицы X вычест
  sstd = X.std(axis=0)
  X = X[:, sstd!=0]   # удалим столбцы, где все числа одина
  X /= X.std(axis=0)  # Каждый столбец матрицы X поделить на

  N,K = X.shape
  M = Y.max()+1
  print(f'N={N}, K={K}, M={M}')
  ...
```

Run продолжение

```
def tst_01():   # общая проверка
  ...
  # добавление столбца из единиц в начало матрицы
  X = np.hstack((np.ones(len(X)).reshape((-1,1)), X))
  W = np.random.normal(0, 1, size=(M,K+1))
  LR =1.0
  k_step = 5      # сколько шагов за раз
  for iStep in range(400):
    res = gradient_descent(f, df, W.reshape(-1), LR,
                           n_step=k_step )
    W = res[0].reshape(W.shape)
    idx = (iStep+1)*k_step
    acc = accuracy(Y,X,W)
    print(f'{idx:4d}, accuracy={acc:8.6f}, S={res[1]:8.5f},
          steps={res[2]:4d}')
    #draw_dim2(Y, X, W, acc=acc, idx=idx)
```