Logistic Regression

January 24, 2018

0.1 Introduction to Logistic Regression

0.1.1 Standard Form of a Line

$$0 = ax + by + c$$

0.1.2 Hypothesis function

$$h(x, y) = a\ function\ in\ x\ and\ y$$

0.1.3 Conversion to Vectors

$$(x, y) \rightarrow (x_1, x_2) = \mathbf{x}$$

Constants in the linear equation are renamed to w_i. The bias term, or intercept is w_0. Vector $\mathbf{x}$ is a vector of weights.

$$h(\mathbf{x}) = \mathbf{w}^{\mathbf{T}}\mathbf{x}$$

0.2 Logistic Function

0.2.1 Notation

$N = Number of samples $ $D = Number of dimensions (features) $ $**X** = N x D matrix $ $**w** = N x 1 matrix of weights $ $h(x) = hypothesis function $ $z = \mathbf{w}^{\mathbf{T}}\mathbf{x}$

https://en.wikipedia.org/wiki/Logistic_function

In logistic regression, this is referred to as the sigmoid.

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

0.2.2 Logistic Function in Vector Form

$$P(y = 1 \mid x) = \sigma(w^T x)$$

0.2.3 Basic Example of Logistic Regression

```
In [41]: import numpy as np

         N = 100
         D = 2

         # Generate NxD matrix with random values.
         # randn pulls random numbers from the normal distribution
         # with mean = 0 and variance = 1
         X = np.random.randn(N,D)
         print(type(X))
         print(X)
```

```
<class 'numpy.ndarray'>
[[ -4.86884944e-01  -3.14214678e-01]
 [  2.63962817e-01  -5.62602492e-01]
 [ -1.66048457e-01   5.31665529e-01]
 [ -2.62903617e-01   8.48151940e-01]
 [  8.08819785e-02   1.40970810e-02]
 [ -5.19676460e-01  -4.83824487e-01]
 [  2.77718516e-01   4.64008472e-01]
 [  2.79140139e-01  -7.91252219e-02]
 [  1.04088748e+00  -2.55707216e-02]
 [ -3.45776297e-01  -9.05005394e-02]
 [  2.38974282e-01   2.06140031e-01]
 [  6.72900877e-01   1.32054779e+00]
 [  5.33245730e-01  -1.16622915e+00]
 [  3.28605815e-01   9.95127801e-01]
 [ -1.12365455e+00  -1.01323597e+00]
 [  1.51254002e+00   9.73109586e-01]
 [ -7.50756142e-01   5.34793982e-01]
 [ -4.39575241e-01   1.20493489e+00]
 [ -3.24318149e-01  -9.64784515e-01]
 [ -1.98789997e+00   5.39503403e-01]
 [ -2.65770564e-01  -2.30488491e+00]
 [  2.63749206e-01   3.45128101e-01]
 [  3.50563620e-01  -1.79865878e-01]
 [ -1.03054772e+00  -9.21329514e-01]
 [  1.05681631e-03  -1.51162972e-01]
 [  1.12620242e+00  -4.52884784e-01]
 [ -4.76663802e-01   1.41433139e-01]
 [ -3.85627846e-02  -2.16089123e-01]
 [  6.20990514e-01   8.84025989e-01]
 [ -1.03461591e-01  -1.07951453e-01]
 [ -7.42477637e-01   4.86025021e-01]
 [ -1.27821788e+00  -1.24215103e+00]
 [  6.93377428e-01  -1.93464540e+00]
 [  1.44629605e+00   9.69132821e-01]
 [ -1.85254067e+00  -9.74955017e-02]
 [  1.96914063e+00   7.81183475e-01]
 [  1.62249554e+00  -7.45120676e-01]
 [  9.31144139e-01   7.27855902e-01]
 [ -1.56085284e-01   3.16714002e-01]
 [  1.49061539e+00   4.38943292e-01]
 [ -2.50696523e-01  -9.35320148e-02]
 [ -4.39652496e-01  -3.17419795e-01]
 [ -1.24815278e+00  -5.02721562e-01]
 [ -3.20264052e-01   9.86613952e-01]
 [ -1.46069940e+00   7.71231015e-02]
 [  2.86706157e-01   1.66036977e-01]
 [  3.43261763e-01   3.14899083e-01]
 [  5.57085833e-01   9.84841428e-01]
 [  3.40690473e-01   1.72735175e-01]
 [  3.25202358e-01  -1.01342614e+00]
 [ -1.72845457e+00  -7.69028378e-02]
 [ -1.38679315e+00   5.45347130e-01]
 [  9.69593585e-01  -4.63974119e-01]
 [  5.18419929e-01   9.53113002e-01]
 [ -4.92846097e-01  -9.25228765e-01]
 [ -1.88677292e-01   1.05776566e+00]
 [  5.33558610e-01   1.04587121e-01]
 [ -1.74492153e-01   1.44496939e-01]
 [ -3.54736747e-02  -7.34750145e-01]
 [ -9.37106597e-01   1.46428870e+00]
 [  1.23772751e+00  -6.54799354e-01]
 [ -1.04383829e-01   3.16933386e+00]
 [  4.66910720e-01  -3.38000766e-01]
 [  2.74812741e-02   1.58247892e+00]
 [ -1.36856006e+00   8.83570601e-01]
 [  1.13442296e+00   1.36581072e+00]
 [ -1.24097788e+00   1.27308340e+00]
 [  1.38169056e+00  -8.31027606e-01]
 [  7.57551874e-01   7.61024565e-01]
 [ -1.48108597e+00  -1.31421194e+00]
 [  7.94299896e-01   1.02423425e+00]
 [  1.61888085e+00   1.40658714e+00]
 [  5.23321108e-01   2.15398909e+00]
 [  1.52894868e+00   9.06240605e-01]
 [ -1.29138702e+00  -4.14053062e-02]
 [  9.61529213e-01   8.90882852e-01]
 [ -1.32555988e+00  -2.59351651e-01]
 [  2.34800736e-01   6.65977324e-01]
 [ -1.80036454e+00   6.35136970e-01]
 [ -6.10318493e-01   9.76050144e-01]
 [ -7.90475524e-01  -5.29862095e-03]
 [  1.01491355e+00   7.39022711e-01]
```

```
 [ -1.06612962e+00   1.28989102e+00]
 [  9.45366574e-01  -1.50105162e+00]
 [  1.17311567e+00  -1.67035701e-01]
 [ -1.99369751e-01  -9.47602699e-01]
 [ -1.83933956e+00   1.73443212e+00]
 [  1.33238600e-03  -1.70472533e-01]
 [ -1.24731825e+00   1.27385901e+00]
 [  9.69469435e-01  -2.96938591e-01]
 [  1.50546370e+00   1.14812333e-01]
 [  2.39147510e-01  -9.80867925e-01]
 [ -3.38557349e-01   1.21976691e+00]
 [ -3.72501276e-01  -1.38872264e+00]
 [ -2.02175948e-01   5.18889496e-01]
 [  3.15664990e-03  -2.27151816e-01]
 [  2.06299607e+00  -9.62508243e-01]
 [  1.08674163e+00  -1.93993909e+00]
 [ -1.42048568e+00   6.91384293e-01]
 [ -1.14268269e-01  -5.17988415e-01]]
```

```
In [ ]: #Add a bias term by
        #(1) Add a column on 1s in the original data.
        #(2) Include the bias in the weights w[0]

        # Transpose a 1xN matrix to get an Nx1 matrix
        ones = np.array([[1]*N]).T
        print(ones)
```

```
In [7]: #Concatenate the vector of 1s to the original dataset to make vector Xb
        Xb = np.concatenate((ones, X), axis = 1)
        print(Xb)
```

```
[[  1.00000000e+00   5.12499476e-01   8.24985293e-01]
 [  1.00000000e+00  -2.97769740e-01  -1.07955968e+00]
 [  1.00000000e+00   2.08987563e-01   8.35141279e-01]
 [  1.00000000e+00  -7.90140861e-01   9.31091348e-01]
 [  1.00000000e+00  -8.97628214e-01  -1.34534890e+00]
 [  1.00000000e+00  -1.11147109e+00  -9.69534125e-04]
 [  1.00000000e+00  -7.64277282e-01  -4.81471441e-01]
 [  1.00000000e+00  -2.03562482e+00   1.81405027e+00]
 [  1.00000000e+00   9.41461078e-01  -1.27002883e+00]
 [  1.00000000e+00  -9.20321929e-01   1.29929415e+00]
 [  1.00000000e+00  -4.54928217e-01  -1.15874836e-01]
 [  1.00000000e+00  -2.10773759e-01   6.70186275e-01]
 [  1.00000000e+00   4.85384512e-01  -3.46263901e-01]
 [  1.00000000e+00  -8.57399212e-01   1.44707497e-01]
 [  1.00000000e+00  -1.98975042e-01   5.47269332e-01]
 [  1.00000000e+00   5.11858314e-01   1.61527159e+00]
[  1.00000000e+00   4.00003108e-02  -1.83812365e+00]
[  1.00000000e+00   3.10198939e-01   3.78186039e-01]
[  1.00000000e+00   3.05680356e-02   7.50654828e-01]
[  1.00000000e+00  -1.00554307e+00  -1.66437728e+00]
[  1.00000000e+00  -2.33027912e+00  -1.21510657e+00]
[  1.00000000e+00   2.45397009e+00   2.27743504e-01]
[  1.00000000e+00   7.64849257e-01  -1.06857182e+00]
[  1.00000000e+00   2.60811860e+00  -9.33918935e-01]
[  1.00000000e+00  -1.06735166e+00   4.92013994e-01]
[  1.00000000e+00  -8.44990250e-01  -1.68899159e+00]
[  1.00000000e+00   1.57803078e+00  -4.08438460e-01]
[  1.00000000e+00  -2.68136414e-01  -1.37265165e+00]
[  1.00000000e+00  -1.70476028e-01   8.72341359e-01]
[  1.00000000e+00  -1.02391577e+00   1.58063936e+00]
[  1.00000000e+00  -2.49060453e+00  -7.48010142e-01]
[  1.00000000e+00  -9.22293791e-01   1.97333060e+00]
[  1.00000000e+00  -1.05456409e-01  -2.07090564e-01]
[  1.00000000e+00  -6.80958212e-01  -1.20913322e+00]
[  1.00000000e+00  -1.10955486e+00   9.19699181e-01]
[  1.00000000e+00  -1.29703287e+00  -1.31037281e-01]
[  1.00000000e+00   1.25253964e+00   6.93481845e-01]
[  1.00000000e+00  -9.54354977e-01   7.00848328e-01]
[  1.00000000e+00  -1.18231376e+00  -2.05515241e-02]
[  1.00000000e+00  -1.10744117e+00   1.97651661e+00]
[  1.00000000e+00  -4.39388353e-01  -2.97690632e-01]
[  1.00000000e+00   2.68358899e-01   2.12110734e-01]
[  1.00000000e+00  -8.80776353e-01  -7.95129496e-01]
[  1.00000000e+00   2.08660629e-01   1.81871763e+00]
[  1.00000000e+00  -4.34439538e-01  -9.33433952e-01]
[  1.00000000e+00   2.11916142e-01  -1.14594247e+00]
[  1.00000000e+00   8.09653628e-01   7.71721697e-01]
[  1.00000000e+00  -1.99423326e-02  -7.63339890e-01]
[  1.00000000e+00   1.86345930e+00   7.33534285e-01]
[  1.00000000e+00  -1.14312927e-01   6.13795756e-01]
[  1.00000000e+00   9.31551764e-01   3.33502457e-01]
[  1.00000000e+00   1.69761782e+00  -5.68121146e-01]
[  1.00000000e+00  -1.80471015e+00   3.04036108e-01]
[  1.00000000e+00   7.07147748e-01  -3.21256977e-01]
[  1.00000000e+00   1.25781293e+00  -8.70199717e-01]
[  1.00000000e+00  -1.51317413e+00  -1.54145977e+00]
[  1.00000000e+00   2.29956229e+00   1.24036781e+00]
[  1.00000000e+00   1.46482227e-01  -1.49100470e+00]
[  1.00000000e+00  -1.42788770e+00  -3.93972438e-01]
[  1.00000000e+00  -1.09573422e+00   4.78593978e-01]
[  1.00000000e+00  -7.78506736e-01   4.53770284e-01]
[  1.00000000e+00   1.00860379e-01  -8.24075400e-01]
[  1.00000000e+00   1.77017914e+00  -1.58102626e+00]
[  1.00000000e+00  -1.72038787e-01  -9.08426913e-01]
```

```
 [  1.00000000e+00   7.82108737e-01  -2.19387800e-02]
 [  1.00000000e+00   1.35237019e+00   5.06697654e-02]
 [  1.00000000e+00  -1.19206387e+00   5.08739270e-01]
 [  1.00000000e+00   3.57102489e-01  -5.52649224e-01]
 [  1.00000000e+00   2.66354921e+00  -4.39752858e-01]
 [  1.00000000e+00   3.13875923e-01   2.39397246e+00]
 [  1.00000000e+00  -8.95108492e-01  -1.45165323e+00]
 [  1.00000000e+00   1.62234223e-01   1.13583073e+00]
 [  1.00000000e+00  -5.18173852e-01  -1.15577388e+00]
 [  1.00000000e+00   7.85674074e-01  -6.29719264e-01]
 [  1.00000000e+00  -1.21990205e-01  -3.00351238e-02]
 [  1.00000000e+00  -1.30119543e+00   8.99156757e-01]
 [  1.00000000e+00   6.77037807e-01  -8.60292817e-01]
 [  1.00000000e+00  -1.03421821e+00  -7.91522848e-01]
 [  1.00000000e+00   6.92391844e-01   6.39343212e-01]
 [  1.00000000e+00   7.67611228e-01   4.73267950e-01]
 [  1.00000000e+00  -2.88147458e-01   5.65989741e-01]
 [  1.00000000e+00  -3.39403693e-01  -4.81543962e-01]
 [  1.00000000e+00   1.91866028e-01  -8.88304849e-01]
 [  1.00000000e+00  -2.54250530e-01   1.46012168e+00]
 [  1.00000000e+00   2.25609904e-01   3.90943128e-02]
 [  1.00000000e+00   2.71155564e-02  -6.66340465e-01]
 [  1.00000000e+00  -6.09296680e-01  -2.62688594e-01]
 [  1.00000000e+00  -4.83304332e-01   3.70549526e-01]
 [  1.00000000e+00   5.00244888e-01   1.61200050e+00]
 [  1.00000000e+00  -3.99715731e-01  -7.02542073e-02]
 [  1.00000000e+00   3.34081392e-01   6.68039303e-01]
 [  1.00000000e+00  -8.29932326e-01  -8.73598430e-01]
 [  1.00000000e+00  -8.15290102e-01  -1.68894766e+00]
 [  1.00000000e+00   2.59396952e+00   9.44457759e-01]
 [  1.00000000e+00  -1.44997772e-01  -2.16179626e-01]
 [  1.00000000e+00  -1.41226817e+00   2.87361599e-02]
 [  1.00000000e+00  -1.24695850e+00  -1.90232855e+00]
 [  1.00000000e+00   8.07007537e-01  -1.69152451e+00]
 [  1.00000000e+00   1.41504278e+00  -1.89118311e+00]
 [  1.00000000e+00  -7.85468842e-01   6.68739522e-01]]
```

```
In [39]: #Randomly initialize a weight vector
         w = np.random.randn(D + 1)
         print(w)
         #w2 = np.random.randn(D + 1, 1)
         #print(w2)
         #print(w2.T)
```

```
[ 0.62035588 -0.10889877 -0.36982741]
```

```
In [40]: #Calculate the dot product between each row of X and w
```

```
        z = Xb.dot(w)
        print(z)
        #z2 = Xb.dot(w2.T)
        #z2 = (w2.T) @ (Xb)
        #z2 = np.matmul(w2.T, Xb)
        #w2.T.dot(Xb)
        #print(z2)
```

```
[ 0.25944314  1.0520334   0.28873925  0.36205814  1.21565339  0.74175227
  0.88164607  0.1711474   0.9875234   0.24006321  0.71275069  0.39545563
  0.69555598  0.6602088   0.43962881 -0.03275657  1.2957884   0.44671203
  0.33941432  1.34539062  1.32350012  0.26889576  0.93225188  0.68172379
  0.5546289   1.33700966  0.599562    1.15719981  0.31630476  0.14729528
  1.1682143  -0.00899921  0.70842772  1.141682    0.40105507  0.81006234
  0.22748726  0.46509104  0.75670891  0.00998484  0.77829888  0.51268756
  1.01033202 -0.07497864  1.01287527  1.02107941  0.24678176  0.90483159
  0.14614636  0.40580592  0.39557268  0.64559416  0.70444571  0.66215799
  0.8052053   1.35521275 -0.08878565  1.15581855  0.9215529   0.56268281
  0.53731761  0.91413798  1.01229239  0.97505186  0.54329876  0.45434536
  0.56202444  0.78585269  0.4929313  -0.29918146  1.25469325  0.18262743
  1.10422123  0.76768438  0.64474827  0.42952164  0.86478716  1.02570781
  0.30850861  0.3617365   0.44241626  0.83540468  0.92798138  0.10805043
  0.58132909  0.86383399  0.78385698  0.53594775 -0.03028215  0.68986636
  0.33691558  1.03381513  1.33375911 -0.01141058  0.71609511  0.76352272
  1.45968137  1.15804588  1.16567081  0.45857426]
```

```
In [16]: def sigmoid(z):
             return 1/(1 + np.exp(-z))
```

```
In [17]: # Results are Nx1
         print(sigmoid(z))
```

```
[ 0.89837892  0.32602634  0.88651374  0.8496258   0.20556824  0.58509855
  0.46509805  0.91257845  0.39659096  0.89693972  0.61951187  0.83833064
  0.64693442  0.6576545   0.81567642  0.96204447  0.17012293  0.81640595
  0.86556333  0.13871823  0.13869862  0.90560853  0.44250901  0.68614663
  0.73515348  0.1435416   0.73436048  0.2490743   0.87364664  0.92357254
  0.21824455  0.95527195  0.62806428  0.2549739   0.8280719   0.52158996
  0.91203285  0.7942026   0.57091707  0.95175007  0.56269618  0.77762058
  0.35190278  0.96664914  0.35589845  0.35796599  0.90402085  0.45537285
  0.93513625  0.83400833  0.84730593  0.70225481  0.60750667  0.67682619
  0.56375786  0.13105233  0.97184128  0.25467604  0.41987009  0.72912767
  0.75036794  0.44884901  0.38710188  0.3914553   0.76353724  0.82154894
  0.72846023  0.56768828  0.81245582  0.98495063  0.18355692  0.91948303
  0.28343223  0.58988801  0.67972119  0.81122135  0.50174363  0.33731717
  0.88217649  0.86126237  0.81335414  0.51305224  0.43787429  0.93579547
  0.73142372  0.49290427  0.55525668  0.75456978  0.96169188  0.63951685
  0.86868852  0.33352065  0.1452081   0.96373565  0.62102973  0.56151288
  0.09514341  0.26068187  0.26238301  0.79963582]
```

0.3 Cross-entropy cost function for binary classification.

Is also the negative log-likelihood of the model outputs.

$J = cost function (error function or objective function) $ $N = samples $ $y = target? $ $Y = short\ form\ of\ P(Y = 1 \mid X)$

$$J = -\sum_{i=1}^{N} t_i log(y_i) + (1 - t_i) log(1 - y_i)$$

0.4 Naive Bayes

http://scikit-learn.org/stable/modules/naive_bayes.html

$$P(y|x_i, ...x_n) = P(y) \prod_{i=1}^{n} P(x_i|y)$$

$$\hat{y} = arg\ max\ P(y) \prod_{i=1}^{n} P(x_i|y)$$