CSE 5523: Lecture Notes 12
Logistic Regression

Contents

12.1 (Multinomial) Logistic Regression 1
12.2 Sample logistic regression code 2

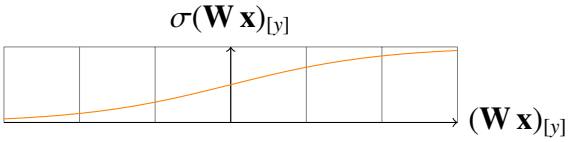
12.1 (Multinomial) Logistic Regression

Sometimes our conditioned-on variables $\mathbf{x}$ are numerical but our modeled variable y is discrete.

We can model this without independence assumptions if we use a **(multinomial) logistic model** (here $y \in \{1, \ldots, K\}$, $\mathbf{W} \in \mathbb{R}^{K \times V}$, $\mathbf{x} \in \mathbb{R}^V$):

$$\mathsf{P}_{\mathbf{W}}(y \mid \mathbf{x}) \;=\; \sigma(\mathbf{W}\,\mathbf{x})_{[y]} \stackrel{\text{def}}{=} \frac{e^{\delta_y^\top \mathbf{W}\,\mathbf{x}}}{\sum_{y'} e^{\delta_{y'}^\top \mathbf{W}\,\mathbf{x}}}$$

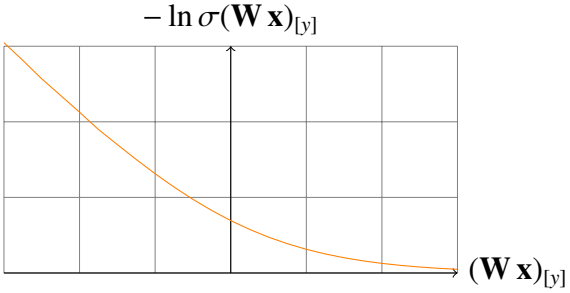
It finds smooth differentiable sigmoid (word-final-sigma-'ς'-shaped) separators for $\{0,1\}^K$ points (the $\{0,1\}^K$ points are one-hot dimensions of a Kronecker delta δ_y).



This is '**multinomial**' if $K > 2$. It is also called a **softmax** or **maximum entropy** model.

We then fit parameters using a **negative log loss** function to make low probabilities linearly bad:

$$\mathsf{L}_{\mathrm{NL}}(y, \sigma(\mathbf{W}\,\mathbf{x})_{[y]}) = -\ln \frac{e^{\delta_y^\top \mathbf{W}\,\mathbf{x}}}{\sum_{y'} e^{\delta_{y'}^\top \mathbf{W}\,\mathbf{x}}}$$



We can then find the slope (derivative) of the expected loss (which is therefore cross entropy):

$$
\begin{aligned}
&\frac{\partial}{\partial \mathbf{W}_{[k,v]}} \overbrace{\frac{1}{N} \sum_{\langle y, \mathbf{x} \rangle \in \mathcal{D}}}^{\text{expectation}} \overbrace{-\ln \frac{e^{\delta_y^\top \mathbf{W}\mathbf{x}}}{\sum_{y'} e^{\delta_{y'}^\top \mathbf{W}\mathbf{x}}}}^{\text{loss}} \\
&= \frac{1}{N} \sum_{\langle y, \mathbf{x} \rangle \in \mathcal{D}} \frac{\partial}{\partial \mathbf{W}_{[k,v]}} - \ln \frac{e^{\delta_y^\top \mathbf{W}\mathbf{x}}}{\sum_{y'} e^{\delta_{y'}^\top \mathbf{W}\mathbf{x}}} && \text{sum rule} \\
&= \frac{1}{N} \sum_{\langle y, \mathbf{x} \rangle \in \mathcal{D}} \frac{\partial}{\partial \mathbf{W}_{[k,v]}} - \left(\ln e^{\delta_y^\top \mathbf{W}\mathbf{x}} - \ln \sum_{y'} e^{\delta_{y'}^\top \mathbf{W}\mathbf{x}} \right) && \text{log of fraction} \\
&= \frac{1}{N} \sum_{\langle y, \mathbf{x} \rangle \in \mathcal{D}} \frac{\partial}{\partial \mathbf{W}_{[k,v]}} - \delta_y^\top \mathbf{W}\mathbf{x} + \ln \sum_{y'} e^{\delta_{y'}^\top \mathbf{W}\mathbf{x}} && \text{log of exponentiation} \\
&= \frac{1}{N} \sum_{\langle y, \mathbf{x} \rangle \in \mathcal{D}} -[\![y{=}k]\!]\, \mathbf{x}^\top \delta_v + \frac{\partial}{\partial \mathbf{W}_{[k,v]}} \ln \sum_{y'} e^{\delta_{y'}^\top \mathbf{W}\mathbf{x}} && \text{sum,product rule} \\
&= \frac{1}{N} \sum_{\langle y, \mathbf{x} \rangle \in \mathcal{D}} -[\![y{=}k]\!]\, \mathbf{x}^\top \delta_v + \left(\frac{1}{\sum_{y'} e^{\delta_{y'}^\top \mathbf{W}\mathbf{x}}} \right) \frac{\partial}{\partial \mathbf{W}_{[k,v]}} \sum_{y'} e^{\delta_{y'}^\top \mathbf{W}\mathbf{x}} && \text{derivative of log} \\
&= \frac{1}{N} \sum_{\langle y, \mathbf{x} \rangle \in \mathcal{D}} -[\![y{=}k]\!]\, \mathbf{x}^\top \delta_v + \left(\frac{e^{\delta_j^\top \mathbf{W}\mathbf{x}}}{\sum_{y'} e^{\delta_{y'}^\top \mathbf{W}\mathbf{x}}} \right) \frac{\partial}{\partial \mathbf{W}_{[k,v]}} \delta_j^\top \mathbf{W}\mathbf{x} && \text{sum,product rule} \\
&= \frac{1}{N} \sum_{\langle y, \mathbf{x} \rangle \in \mathcal{D}} -[\![y{=}k]\!]\, \mathbf{x}^\top \delta_v + \left(\frac{e^{\delta_j^\top \mathbf{W}\mathbf{x}}}{\sum_{y'} e^{\delta_{y'}^\top \mathbf{W}\mathbf{x}}} \right) (\mathbf{x}^\top \delta_v) && \text{product rule} \\
&= -\delta_k^\top \underbrace{\frac{1}{N} \Big(\mathbf{Y}^\top - \underbrace{\exp(\mathbf{W}\mathbf{X}^\top) \operatorname{diag}\Big(\mathbf{1}^\top \exp(\mathbf{W}\mathbf{X}^\top) \Big)^{-1}}_{\text{prediction}} \Big) \mathbf{X}}_{\text{error}} \delta_v && \text{def. of inner product}
\end{aligned}
$$

(where $\mathbf{Y} \in \{0,1\}^{N \times K}$ is a stack of one-hot output rows, $\mathbf{X} \in \mathbb{R}^{N \times V}$ is a stack of numerical inputs).

Bah, we can't find roots here (the optimization surface is non-linear), but we can use this as update!

Start with any $\mathbf{w}^{(0)}$; move in opposite direction of error times active inputs $\mathbf{x}$, to reduce loss:

$$\mathbf{W}^{(i)} = \mathbf{W}^{(i-1)} + \frac{1}{N} \left(\mathbf{Y}^\top - \exp(\mathbf{W}\mathbf{X}^\top) \operatorname{diag}\Big(\mathbf{1}^\top \exp(\mathbf{W}\mathbf{X}^\top) \Big)^{-1} \right) \mathbf{X}$$

This algorithm is called **gradient descent**.

12.2 Sample logistic regression code

Sample logistic regression code in pandas:

```
import sys
import numpy as np
import pandas
```

```
YX = pandas.read_csv( sys.argv[1] )                                    ## read data
N = len(YX)

Y = pandas.get_dummies( YX[YX.columns[0]] )                            ## one-hot y's
X = YX[YX.columns[1:]]
X['line'] = np.ones(N)                                                 ## line inputs
K = len(Y.columns)
V = len(X.columns)

W = pandas.DataFrame( np.random.rand(K,V), Y.columns, X.columns )      ## random weights

for i in range(10):                                                    ## 10 epochs
  W = W + 1/N * ( Y.T - np.exp(W @ X.T)                                ## update rule
                        @ np.linalg.inv( np.diag( np.ones(K).T @ np.exp(W @ X.T) ) ) ) @ X

print(W)                                                               ## print weights
```

Sample input data file 'YX.csv':

```
y,x1,x2,x3,x4
ya,0,1,1,0
no,1,0,1,0
no,0,1,0,1
ya,1,1,1,1
no,1,0,1,0
no,0,1,0,1
```

Output trained weights:

```
          x1        x2        x3        x4      line
no  0.595169 -0.029497  0.291540  0.583139  1.057525
ya  0.294895  1.085882  1.157317  0.050994 -0.313821
```