Logistic Regression

CSCI 447/547 MACHINE LEARNING

Outline

- Math Behind Logistic Regression
- Visualizing Logistic Regression
- Loss Function
  - Minimizing Log Likelihood Function
- Batch/Full Logistic Regression
- Gradient Descent for OLS
- Gradient Descent for Logistic Regression
- Comparing OLS and Logistic Regression
- Multi-Class Logistic Regression Using Softmax

OLS Recap

- Linear regression
  - Predicts continuous and potentially unbounded labels based on given features
  - $\hat{\bar{Y}} = X\bar{B}$
    - B are the coefficients, X is the data matrix
  - The Issue:
    - Unbounded output means we cannot use it with discrete classification

Logistic Regression

- Classification
  - Predicts discrete labels based on given features
- The Setup
  - y = 0 or 1 with probabilities 1-p and p
  - Predict a probability instead of a value
    - Estimate P(y=1|X)

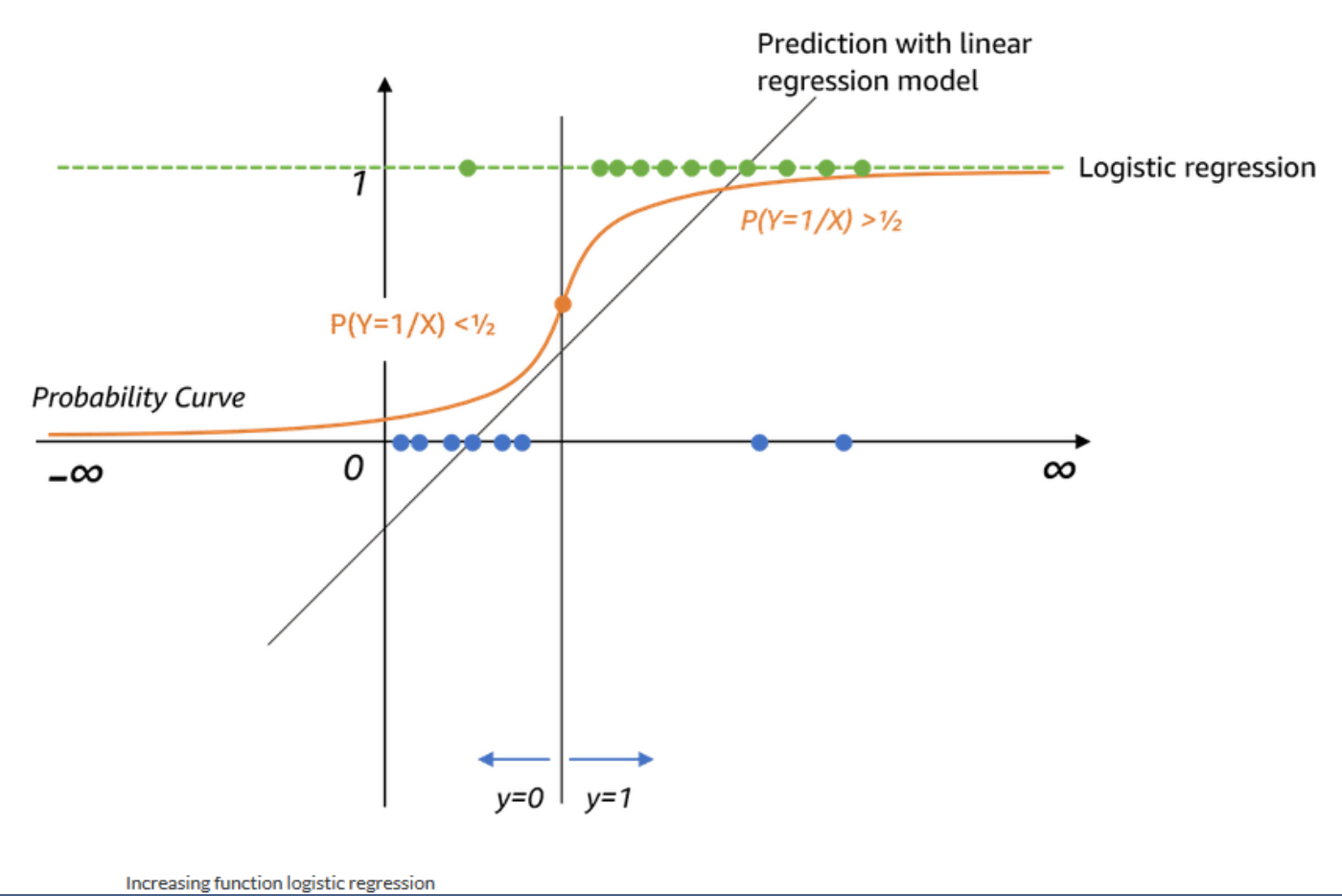
Definitions

- Link Function
  - Relates mean of distribution to output of linear model
    - Convert unbounded to bounded predictions
    - Convert continuous output to discrete interpretation
  - Typically the link function is exponentially distributed
  - Logistic Function
    - Input is ∞ to - ∞ and output is 0 to 1
    - f(- ∞, ∞) -> (0, 1)
    - Typically sigmoid function
      - f(0) = 0.5, f(- ∞) = 0, f(∞) = 1
      - This value is a probability

Note on Choice of Link Function

- Properties:
  - Bounded [0,1]
  - Domain (- ∞, ∞)
  - Differentiable everywhere
    - Used in optimization
  - Increasing function
    - We do not lose the property of coefficient effect (positive and negative) in moving from linear to logistic

Visualizing Logistic Regression



Logistic Regression Loss

- Our Goal
- The Full Log-Likelihood
- A Note on Minimization

Our Goal

- Find P(y=1 | X)
  - Probability to be estimated
- Logistic Function: $\Phi(x) = \frac{1}{1+e^{-x}}$
- OLS Function $\overline{Y} = X\overline{B}$ becomes $\overline{Y} = \Phi(X\overline{B})$
  - $\Phi = \begin{bmatrix} \phi_1 \\ \dots \\ \phi_N \end{bmatrix}$
  - $y_i = \phi(\overline{x_i} \cdot B)$
- MLE – Maximum Likelihood Estimation
  - $L = \prod_{i=1}^{N} P(Y = y_i | \overline{x_i})$
  - Independent, so joint probability is the product of each observation

Our Goal

- L $= \prod_{i=1}^{N} P(Y = \overline{yi}|xi)$
- $= \prod_{i=1}^{N} (P)^{yi}(1-P)^{1\ -\ yi})$
- Log-Likelihood, log(L)
- log(L) $= \sum_{i=1}^{N} [y_i \log p + (1-y_i)\log(1-p)$
- $= \sum_{i=1}^{N} [y_i \log \phi(x_i.B) + (1-y_i)\log(1-\phi(x_i.B))$
- Since we want to minimize, take the negative log likelihood
- -log(L) – will minimize this

The Full Log-Likelihood

$$\mathcal{L}(\vec{\beta}) = -\sum_i [y_i \log(\phi(\vec{x_i} \cdot \vec{\beta})) + (1 - y_i) \log(1 - \phi(\vec{x_i} \cdot \vec{\beta}))]$$

- Minimize this in logistic regression
- So far, similar to OLS
  - But this does not have an explicit solution so we need to minimize this numerically

Logistic Regression Loss

Logistic problem:

Take Linear model $\bar{Y} = X\vec{\beta}$

Apply link model (transformation) $\hat{y}_i = \Phi(\bar{x}_i . \bar{\beta})$

Finding full-log likelihood:

Maximum likelihood function

$$L = \prod_{i=1}^{N} P(Y = y_i | \bar{x}_i)$$

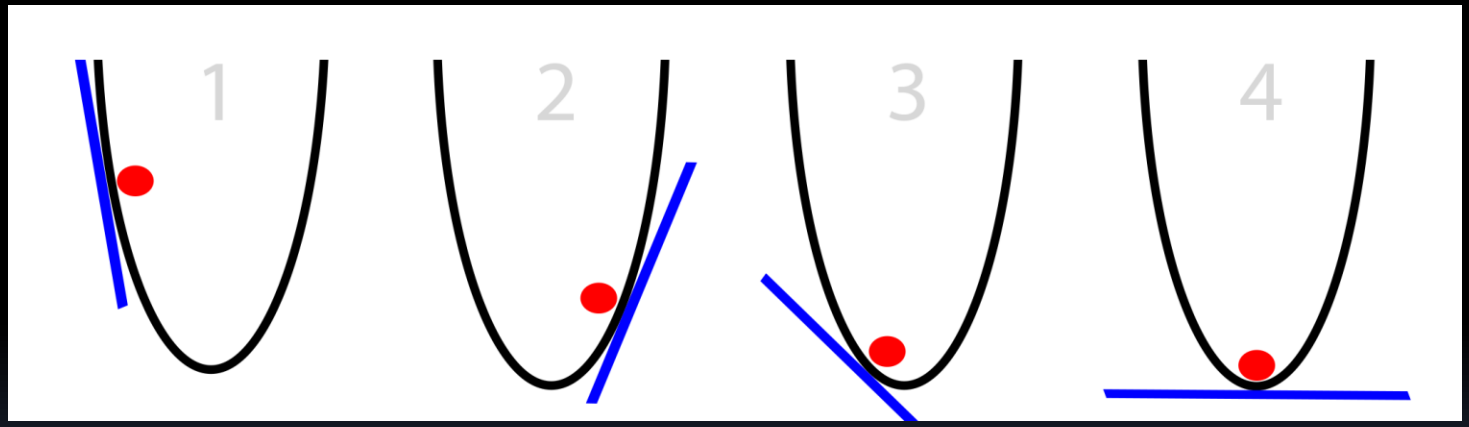
$$\{log(L)\}_{max} = \sum_{i=1}^{N} [y_i \, log \, \Phi(\bar{x}_i . \bar{\beta}) + (1 - y_i) \, log(1 - \Phi(\bar{x}_i . \bar{\beta})]$$

Like OLS method here we find $\{-log(L)\}_{min}$

Minimization:

Minimizing $\{-log(L)\}_{min}$ doesn't have any explicit solutions so we have to minimize it numerically

Gradient Descent



Batch/Full Gradient Descent

- The Gradient
- Algorithm
  - 1. Chose $\bar{x}$ randomly
  - 2. Compute gradient of f at $\bar{x}$: $\nabla f(\bar{x})$
  - 3. Step in the direction of the negative of the gradient
    - $\bar{x} \leftarrow \bar{x} - \eta(\nabla f(\bar{x}))$
    - η = step size (too large, can overshoot, too small, takes too long)
  - Repeat steps 2 and 3 until convergence
    - Difference in iterations is not decreasing or have reached a certain number of iterations

Gradient Descent for OLS

- Numerical Minimization of OLS
  - Mean Log-Likelihood
  - $L = -1/N \, ||\bar{y} - X\bar{B}||_2^2$
- The Algorithm for OLS

The algorithm runs as follows:

- Pick a learning rate η
- Initialize $\vec{\beta}$ with a random guess
- Iterate

$$\vec{\beta} \leftarrow \vec{\beta} - \eta \frac{\partial \mathcal{L}}{\partial \vec{\beta}}$$

$$= \vec{\beta} + \frac{2\eta}{N} X^T [\vec{y} - X\vec{\beta}]$$

Gradient Descent for Logistic Regression

- Numerical Minimization of Logistic Loss

- $L(B) = -\sum_i [y_i \log(\Phi(\bar{x}_i \cdot \bar{B})) + (1-y)\log(1- \Phi(\bar{x}_i \cdot \bar{B}))]$
  - Two Components
  - Recall $\emptyset(x) = \frac{1}{1+e^{-x}}$
  - Because this is symmetric about 1/2:
    - $\emptyset(x) + \emptyset(-x) = 1$

Gradient Descent for Logistic Regression

- First Term
- $\frac{\partial}{\partial B_{j0}}[\log(\Phi(\bar{x}\cdot\bar{B}))] = \frac{1}{\phi(\bar{x}_i\cdot\overline{B})}\frac{\partial}{\partial B_{j0}}\phi(\bar{x}_i\cdot\bar{B})$
- $= \frac{1+e^{-(x_i.B)}}{1}\frac{\partial}{\partial B_{j0}}\left(\frac{1}{1+e^{-(xi.B)}}\right)$
- $= 1+e^{-(xi.B)}\frac{1}{\left(1+e^{-(xi.B)}\right)^2}e^{-xi.B}(-x_{ij})$
- $= \frac{1}{1+e^{xi.B}}x_{ij0}$
- $= \phi(-\bar{x}_i\cdot\overline{B})x_{ij0} = (1-\phi(\bar{x}_i\cdot\overline{B}))x_{ij0}$
- $\frac{\partial}{\partial B_{j0}}[\log(\Phi(\bar{x}\cdot\bar{B}))] = \bar{x}_i^T(1-\phi(\bar{x}_i\cdot\overline{B}))x_{ij0}$

Gradient Descent for Logistic Regression

- Second Term
- $\frac{\partial}{\partial B_{j0}}[\log(1-\Phi(\bar{x}\cdot\bar{B}))] = \frac{\partial}{\partial B}\log(\phi(-\bar{x_i}\cdot\bar{B}))$
- $= -\bar{x_i}^T(1-\phi(-\bar{x_i}\cdot\bar{B}))$
- $= -\bar{x_i}^T(\phi(\bar{x_i}\cdot\bar{B}))$

Gradient Descent for Logistic Regression

- All Terms
- $\frac{\partial L(B)}{\partial \bar{B}} = -\sum_i [\boldsymbol{y_i x_i^T}\,(1-\phi(\bar{x}\cdot\bar{B}))+(1-y_i)(-x_i{}^T)\phi(\bar{x_i}\cdot\bar{B})]$
- $= -\sum_i [\boldsymbol{y_i x_i^T} - y_i x_i{}^T \phi(\bar{x_i}\cdot\bar{B}) + x_i{}^T y_i \phi(\bar{x_i}\cdot\bar{B}) - x_i{}^T \phi(\bar{x_i}\cdot\bar{B})]$
- $= -\sum_i \boldsymbol{x_i^T}\,(y_i - \phi(\bar{x_i}\cdot\bar{B}))$
- $= -\mathrm{X}^{\mathrm{T}}\left(\bar{\boldsymbol{y}} - \boldsymbol{\Phi}(\boldsymbol{X}\bar{\boldsymbol{B}})\right)$
- where $\boldsymbol{\Phi} = \begin{matrix}\phi_1 \\ \dots \\ \phi_N\end{matrix}$

Gradient Descent for Logistic Regression

- $\frac{\partial L}{\partial \bar{B}} = -X^T(\bar{y} - \Phi(X\bar{B}))$
  - where Φ is the individual logistic functions
- Normalize this by the number of observations
  - Divide by N to get the mean loss

Gradient Descent for Logistic Regression

- Algorithm:
  - Pick learning rate η
  - Initialize $\bar{B}$ randomly
  - Iterate:
    - $\bar{B} \leftarrow \bar{B} - \eta \frac{\partial L}{\partial \bar{B}} = \bar{B} + \frac{\eta}{N} X^T [\bar{y} - \Phi(X\bar{B})]$

Comparing OLS and Logistic Regression

- The Two Update Steps
- OLS:
  - $\bar{B} \leftarrow \bar{B} + \frac{2\eta}{N} X^T [\bar{y} - X\bar{B}]$
  - constant residual error
- Logistic:
  - $\bar{B} \leftarrow \bar{B} + \frac{\eta}{N} X^T \left[\bar{y} - \Phi(X\overline{B})\right]$
  - constant error

Comparing OLS and Logistic Regression

- Regularization
- OLS Loss Function under Regularization:
  - $L = \|\bar{y} - X\bar{B}\|_2^2 + \delta^2 \|B\|_2^2$
- Logistic:
  - $L = \left\|\bar{y} - \Phi(X\overline{B})\right\|_2^2 + \delta^2 \|B\|_2^2$

- L2 Norm – Ridge Regression: Uniform Regularization
- L1 Norm – LASSO Regression: Dimensionality Reduction

Multi-Class Logistic Regression Using Softmax

- Softmax
  - $P(y = c_j | x_i) = \frac{e^{\overline{B_j \cdot \overline{x_i}}}}{\sum_{j=1}^{m} e^{\overline{B_j \cdot \overline{x_i}}}}, y \in \{1, \ldots, m\}$
  - $\sum_{j=1}^{m} P(y = c_j | x_i) = 1$, so it is a probability

Multi-Class Logistic Regression Using Softmax

- Comparison with Logistic Regression in the Case of Two Classes
  - 2 Classes: $P(y = 1|x_i) = \frac{1}{1+e^{-x_i\overline{B}}}$
  - Softmax with 2 classes:

    $P(y = 1|x_i) = \frac{e^{x_i\overline{B_1}}}{e^{x_i\overline{B_1}}+e^{x_i\overline{B_2}}}$ multiply this by $\frac{e^{-x_i\overline{B_1}}}{e^{-x_i\overline{B_1}}}$
  - $= \frac{1}{1+e^{-x_i(\overline{B_1}-\overline{B_2})}}$

Softmax Optimization

- Classification Probabilities:
  - $\Pi_{i,j} = \frac{e^{x_i \overline{B_j}}}{\sum_k e^{x_i \overline{B_k}}}$, $\overline{\pi_j} = \begin{matrix} \pi_{1,j} \\ \dots \\ \pi_{N,j} \end{matrix}$
    - Probabilities of observation i, class j
- The Gradients:
  - $\frac{\partial L}{\partial \overline{B_j}} = -X^T\left(\overline{y_j} - \overline{\pi_j}\right)$
    - Logistic: $-X^T(\bar{y} - \phi(X\bar{B}))$
    - Probability vector replaces sigmoid term
    - y_j us a column vector with all entries o except the j^{th}

Softmax Gradient Descent

- Algorithm largely unchanged
- 1. Initialize learning rate
- 2. Randomly choose $B_1 \ldots B_m$
- 3. $\overline{B_j} \leftarrow \overline{B_j} + \frac{\eta}{N} X^T (\overline{y_j} - \overline{\pi_j})$

Last Notes

- Learning Rate η:
  - Small values take a long time to converge
  - Large values, and convergence may not happen
  - Important to monitor loss function on each itertation
- Also need to make sure you normalize so values don't get too large
- Gradient descent algorithms across all are very similar

Summary

- Math Behind Logistic Regression
- Visualizing Logistic Regression
- Loss Function
  - Minimizing Log Likelihood Function
- Batch/Full Logistic Regression
- Gradient Descent for OLS
- Gradient Descent for Logistic Regression
- Comparing OLS and Logistic Regression
- Multi-Class Logistic Regression Using Softmax