Foundations of Data Science, Fall 2020

10. Logistic Regression

Prof. Dan Olteanu

Sept 29, 2020

https://lms.uzh.ch/url/RepositoryEntry/16830890400

https://uzh.zoom.us/j/96690150974?pwd=cnZmMTduWUtCeWoxYW85Z3RMYnpTZz09

Logistic Regression

Discriminative Classification method

- Discriminative: Model the conditional distribution over the output y given the input $\mathbf{x}$ and parameters $\mathbf{w}$

$$p(y \mid \mathbf{w}, \mathbf{x})$$

- Classification: Output y is categorical
  - We first study logistic regression for binary (two classes) classification
    - Today's lecture: We denote the two classes by 0 and 1
    - Future lectures: More convenient to use -1 and $+1$
    - The choice is just for mathematical convenience

$$(-1,+1) \xrightarrow{(y+1)/2} (0,1) \qquad (0,1) \xrightarrow{\mathrm{sign}(y-0.5)} (-1,+1)$$

  - We later discuss multi-class classification

Models for Binary Classification

Bernoulli random variable X takes value in $\{0,1\}$.

$Z \sim \mathrm{Bernoulli}(\theta)$, $\theta \in [0,1]$

$$Z = \begin{cases} 1 & \text{with probability } \theta \\ 0 & \text{with probability } 1-\theta \end{cases}$$

$$p(1 \mid \theta) = \theta$$
$$p(0 \mid \theta) = 1-\theta$$

More succinctly, we can write

$$p(x \mid \theta) = \theta^x (1-\theta)^{1-x}$$

Given input $\mathbf{x}$, models with parameters $\mathbf{w}$ produce a value $f(\mathbf{x}, \mathbf{w}) \in [0,1]$.
We model the (binary) class labels as:

$$y \sim \mathrm{Bernoulli}(f(\mathbf{x}, \mathbf{w}))$$

Logistic Regression

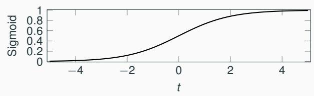
- It builds up on a linear model composed with a sigmoid function

$$p(y \mid \mathbf{w}, \mathbf{x}) = \mathrm{Bernoulli}(\mathrm{sigmoid}(\mathbf{w} \cdot \mathbf{x}))$$

(Wlog $x_0 = 1$, so we do not need to handle the bias term w_0 separately)

- Recall that the sigmoid function σ is defined by:

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



$$\sigma : \mathbb{R} \to (0,1)$$

$$t \geq 0 \Rightarrow \sigma(t) \geq 1/2$$

Prediction Using Logistic Regression

Suppose we have estimated the model parameters $\mathbf{w} \in \mathbb{R}^D$

For a new data point $\mathbf{x}_{\text{new}}$, the model gives us the probability

$$p(y_{\text{new}} = 1 \mid \mathbf{x}_{\text{new}}, \mathbf{w}) = \sigma(\mathbf{w} \cdot \mathbf{x}_{\text{new}}) = \frac{1}{1+\exp(-\mathbf{x}_{\text{new}} \cdot \mathbf{w})}$$

In order to make a prediction we can simply use a threshold at $\frac{1}{2}$

$$\widehat{y}_{\text{new}} = \mathbb{I}(\sigma(\mathbf{w} \cdot \mathbf{x}_{\text{new}})) \geq \frac{1}{2}) = \mathbb{I}(\mathbf{w} \cdot \mathbf{x}_{\text{new}} \geq 0)$$

Class boundary is linear (separating hyperplane)

Side Note: How to Compute Decision Boundary and Contour Lines?

What is the contour line for $p(y = 1 \mid \mathbf{x}, \mathbf{w}) = p_0$?

By definition: $$p(y=1 \mid \mathbf{x}, \mathbf{w}) = \frac{1}{1+\exp(-\mathbf{x} \cdot \mathbf{w})} = \frac{p_0}{1}$$

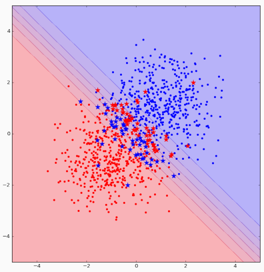
Simplify: $$\frac{1}{1+\exp(-\mathbf{x} \cdot \mathbf{w}) - 1} = \frac{p_0}{1-p_0}$$

Take the log on both sides: $$\log 1 - \log \exp(-\mathbf{x} \cdot \mathbf{w}) = \log \frac{p_0}{1-p_0}$$

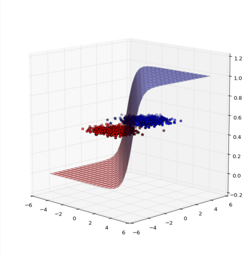
We obtain the hyperplane: $$\mathbf{x} \cdot \mathbf{w} = \log \frac{p_0}{1-p_0}$$

Decision boundary: $p(y = 1 \mid \mathbf{x}, \mathbf{w}) = p(y = 0 \mid \mathbf{x}, \mathbf{w}) = 1/2 \Rightarrow \mathbf{x} \cdot \mathbf{w} = 0$

Contour Lines Represent Class Label Probabilities



- 2D points not linearly separable
- One normal distribution per class
- Contour lines from bottom left to top right: 0.15, 0.3, 0.45, 0.6 ,0.75, 0.9
- Starred points represent mistakes made by the classifier



Likelihood of Logistic Regression

Data $\mathcal{D} = \big((\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)\big)$, where $\mathbf{x}_i \in \mathbb{R}^D$ and $y_i \in \{0, 1\}$

The likelihood of observing the data, given model parameters $\mathbf{w}$:

$$p(\mathbf{y} \mid \mathbf{X}, \mathbf{w}) = \prod_{i=1}^{N} \sigma(\mathbf{w}^\top \mathbf{x}_i)^{y_i} \cdot (1 - \sigma(\mathbf{w}^\top \mathbf{x}_i))^{1-y_i} = \prod_{i=1}^{N} \mu_i^{y_i} \cdot (1 - \mu_i)^{1-y_i}$$

where $\mu_i \stackrel{\text{def}}{=} \sigma(\mathbf{w}^\top \mathbf{x}_i)$

The negative log-likelihood:

$$\mathrm{NLL}(\mathbf{y} \mid \mathbf{X}, \mathbf{w}) = -\sum_{i=1}^{N} (y_i \log \mu_i + (1 - y_i) \log(1 - \mu_i))$$

$\mathrm{NLL}(y_i \mid \mathbf{x}_i, \mathbf{w})$ is the cross-entropy between y_i and μ_i for $y_i \in \{0, 1\}$

Side Note: Entropy

Entropy H is a measure of uncertainty associated with a random variable X

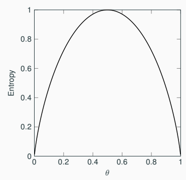
$$H(X) = -\sum_x p(x) \log p(x)$$

- Maximum entropy reached for uniform distributions
- Minimum entropy if all probability mass on one value x

For Bernoulli variable X with parameter θ:

$$H(X) = -\theta \log_2(\theta) - (1 - \theta) \log_2(1 - \theta)$$

Entropy is a useful way to quantify information



Side Note: Cross-Entropy

Let p and q be distributions and suppose the support of p is contained in that of q.

Cross-entropy measures the expected number of bits required to encode an observation from p if the encoding scheme is based on q:

$$H(p, q) = -\sum_x p(x) \log q(x)$$

For our classification: Estimate the probability of different outcomes.

If the estimated probability of outcome i is q_i,

while the frequency (empirical probability) of outcome i in the training set is p_i,

then the negative log-likelihood of the training data is the cross-entropy $H(p, q)$.

The negative log-likelihood for data point $(\mathbf{x}_i, y_i)$:

$$\mathrm{NLL}(y_i \mid \mathbf{x}_i, \mathbf{w}) = -(y_i \log \mu_i + (1 - y_i) \log(1 - \mu_i))$$

is the cross-entropy between y_i and μ_i

Maximum Likelihood Estimate for Logistic Regression: Overview

Recall that $\mu_i = \sigma(\mathbf{w}^\top \mathbf{x}_i)$ and the negative log-likelihood is

$$\mathrm{NLL}(\mathbf{y} \mid \mathbf{X}, \mathbf{w}) = -\sum_{i=1}^{N} (y_i \log \mu_i + (1 - y_i) \log(1 - \mu_i))$$

The gradient with respect to $\mathbf{w}$

$$\nabla_{\mathbf{w}} \mathrm{NLL}(\mathbf{y} \mid \mathbf{X}, \mathbf{w}) = \sum_{i=1}^{N} \mathbf{x}_i (\mu_i - y_i) = \mathbf{X}^\top (\boldsymbol{\mu} - \mathbf{y})$$

The Hessian can be expressed as

$$\mathbf{H} = \mathbf{X}^\top \mathbf{S} \mathbf{X}$$

where $\mathbf{S}$ is a diagonal matrix with $S_{ii} = \mu_i (1 - \mu_i)$

Hessian of NLL is positive definite everywhere ⇔ NLL is convex

We can use convex optimisation methods to minimise NLL

Newton Method for Optimising the Negative Log-Likelihood

For small number D of dimensions, we can apply Newton's method to estimate $\mathbf{w}$

Let $\mathbf{w}_t$ be the parameters after t Newton steps.

The gradient and Hessian are given by:

$$\mathbf{g}_t = \mathbf{X}^\top (\boldsymbol{\mu}_t - \mathbf{y}) = -\mathbf{X}^\top (\mathbf{y} - \boldsymbol{\mu}_t)$$
$$\mathbf{H}_t = \mathbf{X}^\top \mathbf{S}_t \mathbf{X}$$

The Newton update rule:

$$\begin{aligned}
\mathbf{w}_{t+1} &= \mathbf{w}_t - \mathbf{H}_t^{-1} \mathbf{g}_t \\
&= \mathbf{w}_t + (\mathbf{X}^\top \mathbf{S}_t \mathbf{X})^{-1} \mathbf{X}^\top (\mathbf{y} - \boldsymbol{\mu}_t) \\
&= (\mathbf{X}^\top \mathbf{S}_t \mathbf{X})^{-1} (\mathbf{X}^\top \mathbf{S}_t \mathbf{X}) \mathbf{w}_t + (\mathbf{X}^\top \mathbf{S}_t \mathbf{X})^{-1} \mathbf{X}^\top (\mathbf{y} - \boldsymbol{\mu}_t) \\
&= (\mathbf{X}^\top \mathbf{S}_t \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{S}_t \underbrace{(\mathbf{X}\mathbf{w}_t + \mathbf{S}_t^{-1} (\mathbf{y} - \boldsymbol{\mu}_t))}_{\mathbf{z}_t} \\
&= (\mathbf{X}^\top \mathbf{S}_t \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{S}_t \mathbf{z}_t
\end{aligned}$$

Does the above expression for $\mathbf{w}_{t+1}$ look familiar?

From Ordinary Least Squares to Weighted Least Squares

Ordinary Least Squares

$\mathcal{L}(\mathbf{w}) = \sum_i (\mathbf{x}_i^\top \mathbf{w} - y_i)^2$ $\qquad$ $\mathbf{w} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y}$

$\mathcal{L}(\mathbf{w}) = ?$ $\qquad$ $\mathbf{w} = (\mathbf{X}^\top\mathbf{S}_t\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{S}_t\mathbf{z}_t$

$$\mathbf{w} = (\mathbf{X}^\top\mathbf{S}_t^{1/2}\mathbf{S}_t^{1/2}\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{S}_t^{1/2}\mathbf{S}_t^{1/2}\mathbf{z}_t$$

$$\mathbf{w} = (\underbrace{(\mathbf{S}_t^{1/2}\mathbf{X})^\top}_{\tilde{\mathbf{X}}^\top}\underbrace{\mathbf{S}_t^{1/2}\mathbf{X}}_{\tilde{\mathbf{X}}})^{-1}\underbrace{(\mathbf{S}_t^{1/2}\mathbf{X})^\top}_{\tilde{\mathbf{X}}^\top}\underbrace{\mathbf{S}_t^{1/2}\mathbf{z}_t}_{\tilde{\mathbf{y}}}$$

$\mathcal{L}(\mathbf{w}) = \sum_i (\tilde{\mathbf{x}}_i^\top \mathbf{w} - \tilde{y}_i)^2$ $\qquad$ $\mathbf{w} = (\tilde{\mathbf{X}}^\top\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}^\top\tilde{\mathbf{y}}$

$\mathcal{L}(\mathbf{w}) = \sum_i (\mathbf{S}_{t,ii}^{1/2}\mathbf{x}_i^\top \mathbf{w} - \mathbf{S}_{t,ii}^{1/2} z_{t,i})^2$

Weighted Least Squares

$\mathcal{L}(\mathbf{w}) = \sum_i \mathbf{S}_{t,ii}(\mathbf{x}_i^\top \mathbf{w} - z_{t,i})^2$ $\qquad$ $\mathbf{w} = (\mathbf{X}^\top\mathbf{S}_t\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{S}_t\mathbf{z}_t$

Iteratively Re-Weighted Least Squares (IRLS)

We can use weighted least squares to compute $\mathbf{w}_{t+1}$ at each Newton step

- Each step requires re-weighting of the residual by a new diagonal matrix $\mathbf{S}$
- Each step uses a new vector $\mathbf{z}_t$, which depends on $\mathbf{w}_t$
- We proceed iteratively, one Newton step after the other

This optimisation method is called Iteratively Re-Weighted Least Squares

Multi-Class Logistic Regression

Consider now $C > 2$ classes: $y \in \{1, \dots, C\}$

- There are parameters $\mathbf{w}_c \in \mathbb{R}^D$ for every class c
- The parameters form a matrix $\mathbf{W} = [\mathbf{w}_1, \dots, \mathbf{w}_C] \in \mathbb{R}^{D\times C}$
- The multi-class logistic model is given by:

$$p(y = c \mid \mathbf{x}, \mathbf{W}) = \frac{\exp(\mathbf{w}_c^\top \mathbf{x})}{\sum_{c'=1}^{C} \exp(\mathbf{w}_{c'}^\top \mathbf{x})}$$

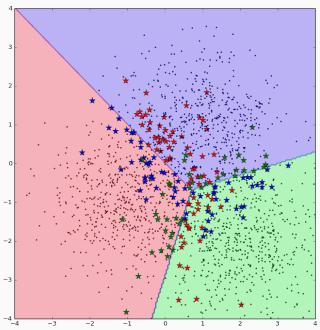
- Parameter estimation: NLL convex, convex optimisation (like in binary case)
- Alternatively expressed using softmax:

$$p(y \mid \mathbf{x}, \mathbf{W}) = \mathrm{softmax}\left(\left[\mathbf{w}_1^\top\mathbf{x}, \dots, \mathbf{w}_C^\top\mathbf{x}\right]^\top\right)$$

- Two-class logistic regression is a special case ($\mathbf{W} = [\mathbf{w}_0, \mathbf{w}_1]$):

$$\mathrm{softmax}\left(\left[\mathbf{w}_1^\top\mathbf{x}, \mathbf{w}_0^\top\mathbf{x}\right]^\top\right)_1 = \frac{\exp(\mathbf{w}_1^\top\mathbf{x})}{\exp(\mathbf{w}_1^\top\mathbf{x}) + \exp(\mathbf{w}_0^\top\mathbf{x})} = \sigma((\mathbf{w}_1 - \mathbf{w}_0)^\top\mathbf{x})$$

Multi-Class Logistic Regression: Decision Boundaries are Linear



- Class red: Data drawn from $\mathcal{N}(\mu_1 = (-1, -1), \sigma^2 = 1)$
- Class blue: Data drawn from $\mathcal{N}(\mu_2 = (1, 1), \sigma^2 = 1)$
- Class green: Data drawn from $\mathcal{N}(\mu_3 = (2, -2), \sigma^2 = 1)$

Summary: Logistic Regression

- Logistic Regression is a (binary) discriminative classification model
- Extension to multiclass by replacing sigmoid with softmax
- Can derive Maximum Likelihood Estimates using Convex Optimisation
- See more in Murphy Section 8.3 (for multi-class)
- Practical 2: Generative vs discriminative models for classification

Basis expansion and regularisation

- Applicable to logistic regression as well
- Regularisation may be necessary if data is linearly separable – Exercise!
- What if the classification boundaries are non-linear?
  - Polynomial or kernel-based basis expansion
  - ℓ_1/ℓ_2 regularisation if risk of overfitting