COMP551 Logistic Regression Tutorial

Lulan Shen

McGill University

lulan.shen@mail.mcgill.ca

September 30, 2022

TA Information

- TA: Lulan Shen, Ph.D. candidate in ECE department, supervised by Prof. James Clark.
- Email: lulan.shen@mail.mcgill.ca
- Office Hour: Fridays 2:00-3:00pm.
  The second tutorial offered by me will cover the regularization topic and be held on Oct. 7th, 1:00-2:00pm.
  Zoom link: https://mcgill.zoom.us/j/84610196676
- TA Dylan Mann-Krzisnik and me are responsible for all the material covered in Assignment 2.

A machine learning system for classification

- A feature representation of the input.
- A classification function such as **sigmoid** and **softmax** that computes $\hat{y}$, the estimated class, via $p(y|x)$.
- An objective function for learning, usually involving minimizing error on training examples. For example, the **cross-entropy (CE) loss function**.
- An algorithm for optimizing the objective function. For example, the **stochastic gradient descent (SGD)** algorithm.

High-Level Views of Binary Classification

- **Probabilistic:**
  - Goal: Estimate $P(y|x)$, i.e. the conditional probability of the target variable given the feature data.
  - Examples: logistics regression, one of the baseline supervised machine learning algorithms for **classification**.
- Decision boundaries:
  - Goal: Partition the feature space into different regions, and classify points based on the region where they lie.
  - Examples: decision trees, support vector machine (SVM), etc.

Approaches to Binary Classification

Two probabilistic approaches:

- **Discriminative learning:**
  - Directly estimate $P(y|x)$.
  - Example: logistic regression.
- Generative learning:
  - Separately model $P(x|y)$ and $P(y)$. Use Bayes' rule, to estimate $P(y|x)$:

$$P(y=1|x) = \frac{P(x|y=1)P(y=1)}{P(x)} \tag{1}$$

  - Example: Linear discriminant analysis (LDA), Naive Bayes, etc.

Probabilistic View of Discriminative Learning

Suppose we have 2 classes: $y \in \{0, 1\}$ what is $P(y = 1|x)$?

$$
\begin{aligned}
P(y=1|x) &= \frac{P(x|y=1)P(y=1)}{P(x)} && (2)\\
&= \frac{P(x|y=1)P(y=1)}{P(x|y=1)P(y=1)+P(x|y=0)P(y=0)} && (3)\\
&= \frac{1}{1+\frac{P(x|y=0)P(y=0)}{P(x|y=1)P(y=1)}} && (4)\\
&= \frac{1}{1+e^{\ln\frac{P(x|y=0)P(y=0)}{P(x|y=1)P(y=1)}}} && (5)\\
&= \frac{1}{1+e^{-a}} = \sigma(a) \Leftarrow \text{logistic function}, && (6)
\end{aligned}
$$

where $a = \ln\frac{P(x|y=1)P(y=1)}{P(x|y=0)P(y=0)} = \ln\frac{P(x|y=1)P(y=1)/P(x)}{P(x|y=0)P(y=0)/P(x)} = \ln\frac{P(y=1|x)}{P(y=0|x)}$ and a is called log-odds ratio.

Probabilistic View of Discriminative Learning

- Log-odds ratio a: How much more likely is $y = 1$ compared to $y = 0$?

$$a = \ln \frac{P(y = 1|x)}{P(y = 0|x)} \tag{7}$$

- Idea: Directly model the log-odds with a linear function of the input feature x.

$$a = \ln \frac{P(y = 1|x)}{P(y = 0|x)} = w_0 + w_1 x_1 + \cdots + w_m x_m = \boldsymbol{w}^T \boldsymbol{x} \tag{8}$$

Since weights are real-valued, the output might even be negative; a ranges from $-\infty$ to ∞.

Decision Boundary

$$\text{Log-odd ratio:} \quad a = \ln \frac{P(y=1|x)}{P(y=0|x)} \tag{9}$$

$$\text{Logistic function:} \quad \sigma(a) = \frac{1}{1+e^{-a}} \tag{10}$$

The **decision boundary** is the set of points for which the linear model predicts zero, i.e. $a = \boldsymbol{w}^T \boldsymbol{x} = 0$.

- If $a = 0$ or $\sigma = 0.5$, Class $y = 1$ is equally likely as Class $y = 0$.
- If $a > 0$ or $\sigma > 0.5$, Class $y = 1$ is more likely than Class $y = 0$.
- If $a < 0$ or $\sigma < 0.5$, Class $y = 1$ is less likely than Class $y = 0$.

Often have a trade-off between false positives and false negatives. Consider logistic regression. Instead of comparing log-odds ratio with zero (threshold=0), vary the threshold.

To build the ROC curve:

1. Train a classifier.
2. Vary the decision boundary threshold.
3. Compute FP rate and TP rate for different decision boundaries associated to the thresholds.

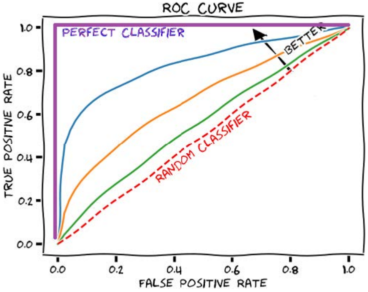

Figure: ROC curve.

Discriminative Learning: Logistic Regression

Sigmoid function $\sigma(\boldsymbol{w}^T\boldsymbol{x})$: What is our predicted probability for $y = 1$?
The **linear logistic/sigmoid function**:

$$\hat{P}(y = 1|x) = \sigma(\boldsymbol{w}^T\boldsymbol{x}) = \frac{1}{1 + e^{-\boldsymbol{w}^T\boldsymbol{x}}} \in [0, 1] \tag{11}$$

$$\hat{P}(y = 0|x) = 1 - \sigma(\boldsymbol{w}^T\boldsymbol{x}) \tag{12}$$

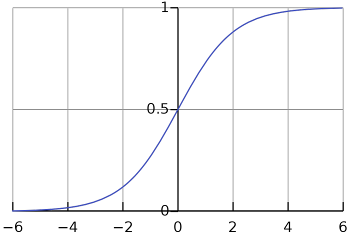

Figure: The sigmoid function takes a real value and maps it to the range [0,1].

Sigmoid Function

Properties:

- $\sigma(a)$ is differentiable.
- $1 - \sigma(a) = \sigma(-a)$

Proof.

$$1 - \sigma(a) = 1 - \frac{1}{1+e^{-a}} = \frac{e^{-a}}{1+e^{-a}} = \frac{e^{-a}}{1+e^{-a}} \cdot \frac{e^{a}}{e^{a}} = \frac{1}{1+e^{a}} = \sigma(-a) \tag{13}$$

$\square$

Since there are only two discrete outcomes (y, which is 1 or 0) of the classifier output ($\hat{y} = P(y = 1|\boldsymbol{x}) = \sigma(\boldsymbol{w}^T \boldsymbol{x})$), this is a Bernoulli distribution.

For a single observation $\boldsymbol{x_i}$, $\hat{y}_i = \sigma(\boldsymbol{w}^T \boldsymbol{x_i})$ we can express

$$p(y_i|\boldsymbol{x_i}) = \hat{y}_i^{y_i}(1 - \hat{y}_i)^{1-y_i} \tag{14}$$

$$= \begin{cases} \sigma(\boldsymbol{w}^T \boldsymbol{x_i}) & \text{if} \quad y_i = 1 \\ 1 - \sigma(\boldsymbol{w}^T \boldsymbol{x_i}) & \text{if} \quad y_i = 0 \end{cases} \tag{15}$$

Maximizing Log-likelihood

The **likelihood** function L describes the joint probability of the observed data as a function of the parameters of the chosen statistical model, and assuming all the observations in the sample are **i.i.d. (independently Bernoulli distributed)**, then

$$
\begin{aligned}
L &= P(y_1, y_2, \ldots, y_n | \boldsymbol{x_1}, \boldsymbol{x_2}, \cdots, \boldsymbol{x_n}, \boldsymbol{w}) && (16)\\
&= \prod_{i=1}^{n} P(y_i | \boldsymbol{x_i}) && (17)\\
&= \prod_{i=1}^{n} \hat{y}_i^{y_i} (1 - \hat{y}_i)^{1-y_i} && (18)
\end{aligned}
$$

Maximizing Log-likelihood

$$\text{Likelihood:} \quad L(\boldsymbol{w}) = \prod_{i=1}^{n} \hat{y}_i^{\,y_i}(1-\hat{y}_i)^{1-y_i} \tag{19}$$

Goal: we choose the parameters $\boldsymbol{w}$ that maximize the probability of the true y labels in the training data given the observations $\boldsymbol{x}$. This is $\boldsymbol{w}^* = \arg\max_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w})$.

Problem: Taking products of lots of small numbers is numerically unstable, making this function hard to optimize.

Solution: Make it easier to optimize by using log-likelihood.

$$\mathcal{L}(\boldsymbol{w}) = \ln(L) = \sum_{i=1}^{n} y_i \ln \hat{y}_i + (1-y_i)\ln(1-\hat{y}_i) \tag{20}$$

Maximizing Likelihood vs. Minimizing Loss

- Another view: The **negative log-likelihood** of the logistic function is known as the **cross-entropy loss**.
- So maximizing the likelihood is the same as minimizing the cross-entropy loss.

$$CE(\boldsymbol{w}) = -\mathcal{L}(\boldsymbol{w}) = -\left(\sum_{i=1}^{n} y_i \ln \hat{y}_i + (1 - y_i) \ln(1 - \hat{y}_i) \right) \tag{21}$$

Note that

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} CE(\boldsymbol{w}) \tag{22}$$

Gradient Descent of Logistic Regression

For logistic regression, this loss function is conveniently convex. A convex function has just one minimum; there are no local minima to get stuck in, so gradient descent starting from any point is guaranteed to find the minimum.

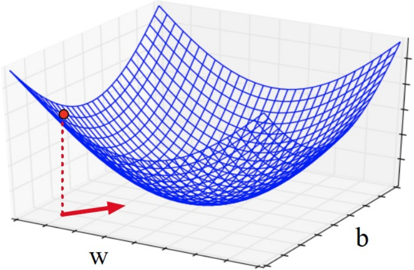

Figure: Visualization of the gradient vector at the red point in two dimensions w and b, showing a red arrow in the x-y plane pointing in the direction we will go to look for the minimum: the opposite direction of the gradient (recall that the gradient points in the direction of increase not decrease).

Gradient Descent for Logistic Regression

Given a random initialization of w at some value w^1 (which is 0 in the figure below), and assuming the loss function CE happened to have the following shape:

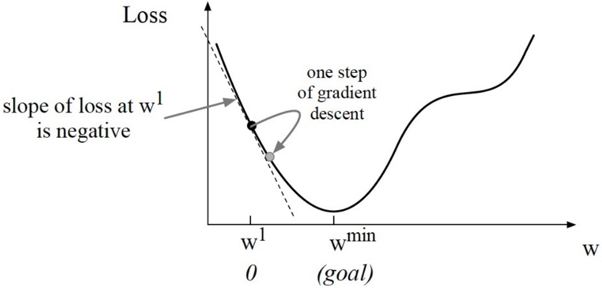

Figure: The first step in iteratively finding the minimum of this loss function, by moving w in the reverse direction from the slope of the function.

Since the slope is negative, we need to move w in a positive direction, to the right, making the loss at w^2 smaller than w^1.

Gradient Descent for Logistic Regression

$$CE(\boldsymbol{w}) = -\mathcal{L}(\boldsymbol{w}) = -\left(\sum_{i=1}^{n} y_i \ln \hat{y}_i + (1 - y_i) \ln(1 - \hat{y}_i)\right) \tag{23}$$

- Take the derivative: (see deviation in lecture7 pp12-14)

$$\frac{\partial CE(\boldsymbol{w})}{\partial \boldsymbol{w}} = -\sum_{i=1}^{n} (y_i - \hat{y}_i)\boldsymbol{x_i} \tag{24}$$

- Update rule:

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \alpha_k \frac{\partial CE(\boldsymbol{w}_k)}{\partial \boldsymbol{w}_k} \tag{25}$$

$$= \boldsymbol{w}_k + \alpha_k \sum_{i=1}^{n} (y_i - \hat{y}_i)\boldsymbol{x_i}, \tag{26}$$

where parameter $\alpha_k \in (0, 1)$ is the learning rate (or step size) or iteration k.

Gradient Descent for Logistic Regression

We want to produce a sequence of weight solutions, $\boldsymbol{w}_0, \boldsymbol{w}_1, \boldsymbol{w}_2, \ldots$, such that: $CE(\boldsymbol{w}_0) > CE(\boldsymbol{w}_1) > CE(\boldsymbol{w}_2) > \ldots$

The algorithm:

1. Given an initial weight vector $\boldsymbol{w}_0$
2. Calculate $\frac{\partial CE(\boldsymbol{w}_k)}{\partial \boldsymbol{w}_k}$, for $k = 0, 1, 2, \ldots$
3. $\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \alpha_k \frac{\partial CE(\boldsymbol{w}_k)}{\partial \boldsymbol{w}_k}$
4. End when $|\boldsymbol{w}_{k+1} - \boldsymbol{w}_k| < \epsilon$

Multinomial Logistic Regression

Sometimes we need more than two classes. In such cases we use **multinomial logistic regression**, also called **softmax regression**.

- The softmax function takes a vector $\boldsymbol{z} = [\boldsymbol{z_1}, \boldsymbol{z_2}, \ldots, \boldsymbol{z_K}]$ of K arbitrary values and maps them to a probability distribution, with each value in the range (0,1), and all the values summing to 1.
- Softmax is defined as:

$$\text{softmax}(\boldsymbol{z_i}) = \frac{e^{\boldsymbol{z_i}}}{\sum_{j=1}^{K} e^{\boldsymbol{z_j}}}, \quad 1 \leq i \leq K \tag{27}$$

$$\text{softmax}(\boldsymbol{z}) = \left[\frac{e^{\boldsymbol{z_1}}}{\sum_{j=1}^{K} e^{\boldsymbol{z_j}}}, \frac{e^{\boldsymbol{z_2}}}{\sum_{j=1}^{K} e^{\boldsymbol{z_j}}}, \ldots, \frac{e^{\boldsymbol{z_K}}}{\sum_{j=1}^{K} e^{\boldsymbol{z_j}}} \right] \tag{28}$$

The denominator $\sum_{j=1}^{K} e^{\boldsymbol{z_j}}$ is used to normalize all the values into probabilities.

Softmax Function

$$\text{softmax}(\boldsymbol{z}) = \left[\frac{e^{z_1}}{\sum_{j=1}^{K} e^{z_j}}, \frac{e^{z_2}}{\sum_{j=1}^{K} e^{z_j}}, \cdots, \frac{e^{z_K}}{\sum_{j=1}^{K} e^{z_j}} \right]$$

Thus for example given a vector:

$$\boldsymbol{z} = [0.6, 1.1, -1.5, 1.2, 3.2, -1.1],$$

the resulting (rounded) softmax($\boldsymbol{z}$) is

$$[0.055, 0.090, 0.006, 0.099, 0.74, 0.010].$$

Note: using natural exponential hugely increases the probability of the highest element and decreases the probability of the lower element when compared with standard normalization.

References

- Armanfard N., ECSE551 Lecture Notes, Montreal, Canada, 2020
- Jurafsky, D.; and Martin, J. H. 2009. Speech and Language Processing (2nd Edition). USA: Prentice-Hall, Inc. ISBN 0131873210.

The End