Machine Learning I (DATS 6202)

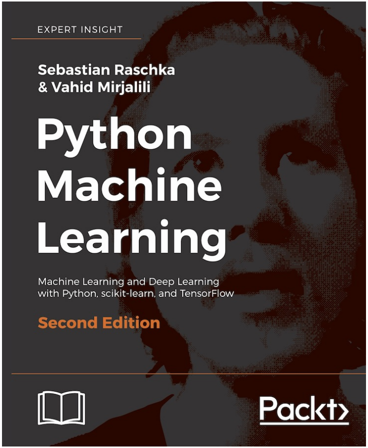
Logistic Regression

Yuxiao Huang

Data Science, Columbian College of Arts & Sciences
George Washington University
yuxiaohuang@gwu.edu

September 18, 2018

Reference



Picture courtesy of the website of the book code repository and info resource

Reference

- This set of slices is an excerpt of the book by Raschka and Mirjalili, with some trivial changes by the creator of the slides
- Please find the reference to and website of the book below:
  - *Raschka S. and Mirjalili V. (2017). Python Machine Learning. 2nd Edition.*
  - `https://sebastianraschka.com/books.html`
- Please find the website of the book code repository and info resource below:
  - `https://github.com/rasbt/python-machine-learning-book-2nd-edition`

Overview

Example: applying for a credit card

Credit Score	Approve
750	yes
700	no
...	...
650	yes
760	yes

Example: applying for a credit card

Credit Score	Approve
750	yes
700	no
...	...
650	yes
760	yes
680	?

Example: applying for a credit card

Credit Score	Approve
750	yes
700	no
...	...
650	yes
760	yes
680	$P(\text{yes}\vert 680)$

Problem statement

- Given:
  - Credit score: x
  - Approve: y
- Predict:
  - Probability of y given x: $P(y|x)$

Linear regression?

- Linearity assumption:

$$P(y|x) = w_0 + w_1 x$$

Linear regression?

- Linearity assumption:

$$P(y|x) = w_0 + w_1 x$$

- Any problems?

Linear regression?

- Linearity assumption:

$$P(y|x) = w_0 + w_1 x$$

- Any problems?

$$\underbrace{P(y|x)}_{[0,1]} = \underbrace{w_0 + w_1 x}_{[-\infty,+\infty]}$$

Solution

- Linearity assumption:

$$\underbrace{P(y|x)}_{[0,1]} = \underbrace{w_0 + w_1 x}_{[-\infty,+\infty]}$$

Solution

- Linearity assumption:

$$\underbrace{P(y|x)}_{[0,1]} = \underbrace{w_0 + w_1 x}_{[-\infty,+\infty]}$$

- Find a function f such that

$$\underbrace{f\left(P(y|x)\right)}_{[-\infty,+\infty]} = \underbrace{w_0 + w_1 x}_{[-\infty,+\infty]}$$

The *odds* function

- Find a function f such that

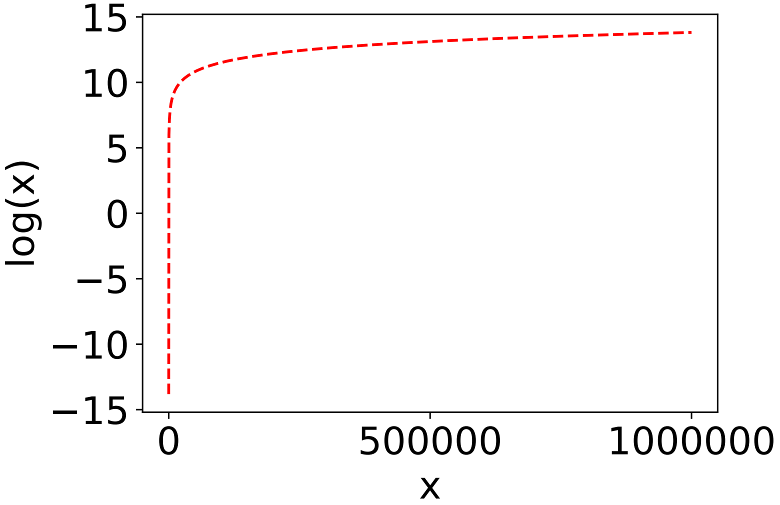
$$\underbrace{f(P)}_{[-\infty,+\infty]} = \underbrace{w_0 + w_1 x}_{[-\infty,+\infty]}$$

- The *odds* function:

$$odds(P) = \frac{P}{\underbrace{1-P}_{[0,+\infty]}}$$

- What else can we do?

Figure 1



The *log odds* function

- Find a function f such that

$$\underbrace{f(P)}_{[-\infty,+\infty]} = \underbrace{w_0 + w_1 x}_{[-\infty,+\infty]}$$

- The *odds* function:

$$odds(P) = \underbrace{\frac{P}{1-P}}_{[0,+\infty]}$$

- The *log odds* function

$$log\ odds(P) = \underbrace{log\left(\frac{P}{1-P}\right)}_{[-\infty,+\infty]}$$

The $logit$ function

- Find a function f such that

$$\underbrace{f(P)}_{[-\infty,+\infty]} = \underbrace{w_0 + w_1 x}_{[-\infty,+\infty]}$$

- The $odds$ function:

$$odds(P) = \underbrace{\frac{P}{1-P}}_{[0,+\infty]}$$

- The $\underbrace{log\ odds}_{logit}$ function

$$\underbrace{log\ odds}_{logit}(P) = \underbrace{log\left(\frac{P}{1-P}\right)}_{[-\infty,+\infty]}$$

The $logit$ function

- The $logit$ function takes input values in the range 0 to 1 and transforms them to values over the entire real number range

$$logit \underbrace{(P(y|x))}_{[0,1]} = \underbrace{log\left(\frac{P(y|x)}{1 - P(y|x)}\right)}_{[-\infty,+\infty]}$$

- Here $P(y|x)$ is the probability that a certain sample belongs to a particular class

Logistic regression

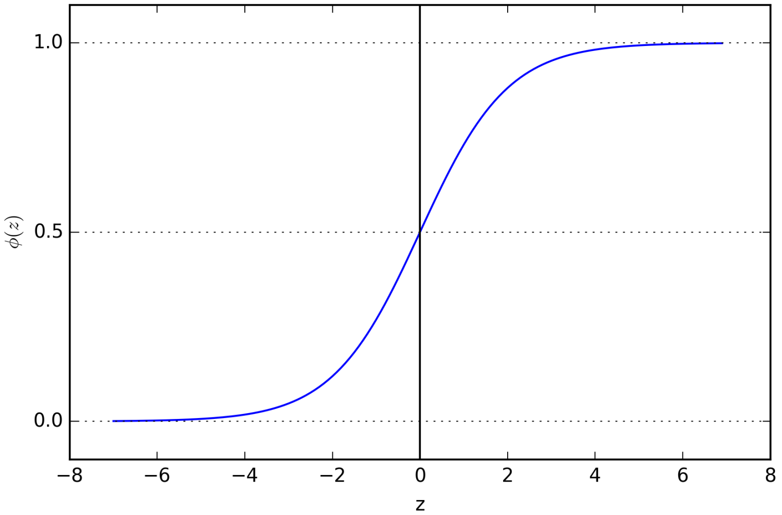
- Logistic regression expresses a linear relationship between the $logit$ and feature values

$$\underbrace{logit\left(P(y|x)\right)}_{[-\infty,+\infty]} = \underbrace{w_0x_0 + w_1x_1 + \cdots + x_mw_m}_{[-\infty,+\infty]} = \sum_{i=0}^{m} w_ix_i = \mathbf{w}^T\mathbf{x}$$

- The goal is predicting $P(y|x)$ using the inverse of the $logit$ function

$$P(y|x) = logit^{-1}\left(\mathbf{w}^T\mathbf{x}\right)$$

Figure 2



The *logistic* function

- The inverse of the *logit* function, $logit^{-1}$, is also called the *logistic* function, sometimes simply abbreviated as the *sigmoid* function (due to its characteristic S-shape), shown in Figure 2 (see previous page)

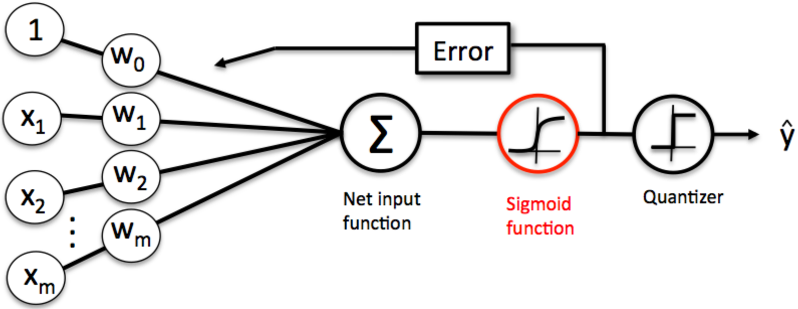
$$\phi(z) = \frac{1}{1 + e^{-z}}$$

- Here, z is the net input, that is, the linear combination of weights and sample features and can be calculated as

$$z = w_0 x_0 + w_1 x_1 + \cdots + x_m w_m = \sum_{i=0}^{m} w_i x_i = \mathbf{w}^T \mathbf{x}$$

- The logistic regression model is shown in Figure 3 (see next page)
- See details in `ch3.ipynb`

Figure 3


1
w_0
x_1
w_1
x_2
w_2
x_m
w_m
Error
Net input function
Sigmoid function
Quantizer
$\hat{y}$

The quantizer

- The predicted probability (the outcome of the sigmoid function) can simply be converted into a binary outcome via a quantizer (unit step function)

$$\hat{y} = \begin{cases} 1 & \text{if } \phi(z) \geq 0.5 \\ 0 & \text{otherwise} \end{cases}$$

- This is equivalent to the following

$$\hat{y} = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

- In fact, there are many applications where we are interested in not only the predicted class labels, but also the probability

Learning the weights of the logistic cost function

- The joint likelihood $L(D|\mathbf{w})$, assuming that the individual likelihood of each sample are independent, is

$$L(D|\mathbf{w}) = \prod_{i=1}^{n} P\big(y^{(i)}|x^{(i)};\mathbf{w}\big) = \prod_{i=1}^{n} \bigg(\phi\big(z^{(i)}\big)\bigg)^{y^{(i)}} \bigg(1 - \phi\big(z^{(i)}\big)\bigg)^{1-y^{(i)}}$$

- The parameters, $\mathbf{w}$, can then be estimated by maximizing the joint likelihood, $L(D|\mathbf{w})$:

$$\hat{\mathbf{w}} = \arg\max_{\mathbf{w}} L(D|\mathbf{w})$$

- This approach is called Maximum Likelihood Estimation (MLE)
- MLE can be solved using optimization algorithms such as gradient ascent (or gradient descent when rewriting the likelihood as a cost function and minimizing it)

The log trick

- In practice, it is easier to maximize the (natural) log of this equation, which is called the log-likelihood function:

$$\hat{\mathbf{w}} = \arg\max_{\mathbf{w}} \left(\log L(D|\mathbf{w})\right)$$

- Here the log-likelihood function is

$$\log\left(L(D|\mathbf{w})\right) = \sum_{i=1}^{n} \left[y^{(i)} \log\left(\phi\left(z^{(i)}\right)\right) + \left(1 - y^{(i)}\right) \log\left(1 - \phi\left(z^{(i)}\right)\right) \right]$$

- The log trick can
  - reduce the potential for numerical underflow (when the likelihoods are very small)
  - simplify the derivation (by converting the product of factors into a summatmation of factors)

Estimating the parameters with gradient descent

- The parameters of the model can be approximated by minimizing the cost function (the additive inverse of the log-likelihood)

$$\begin{aligned} J(w) &= -\log\left(L(D|\mathbf{w})\right) \\ &= -\sum_{i=1}^{n}\left[y^{(i)}\log\left(\phi\left(z^{(i)}\right)\right) + \left(1-y^{(i)}\right)\log\left(1-\phi\left(z^{(i)}\right)\right)\right] \end{aligned}$$

- As in linear regression, here we can minimize the cost function to learn the weights via Gradient Descent (GD)
- Using GD, the rule for updating the weights can be written as

$$\mathbf{w} = \mathbf{w} + \Delta\mathbf{w} \quad \text{where} \quad \Delta\mathbf{w} = -\eta\nabla J(\mathbf{w}).$$

Here:

- η is the learning rate
- $\nabla J(\mathbf{w})$ the gradient of $J(\mathbf{w})$

The updating rule

- The rule for updating w_j can be written as

$$w_j = w_j + \Delta w_j \quad \text{where} \quad \Delta w_j = \eta \sum_i \left(y^{(i)} - \phi\big(z^{(i)}\big) \right) x_j^i.$$

- **Q:** Why does the updating rule work?

The updating rule

- The rule for updating w_j can be written as

$$w_j = w_j + \Delta w_j \quad \text{where} \quad \Delta w_j = \eta \sum_i \Big(y^{(i)} - \phi\big(z^{(i)}\big) \Big) x_j^i.$$

- **Q:** Why does the updating rule work?
- **A:** Because it pulls the predicted probability $(\phi(z))$ closer to the actual one (y)
- Assume $y^{(i)} = 1$, $\phi\big(z^{(i)}\big) = 0.7$, and $x_j^{(i)} > 0$, then:
  1. $\Delta w_j > 0$
  2. $w_j \uparrow$
  3. $\phi(z) \uparrow$

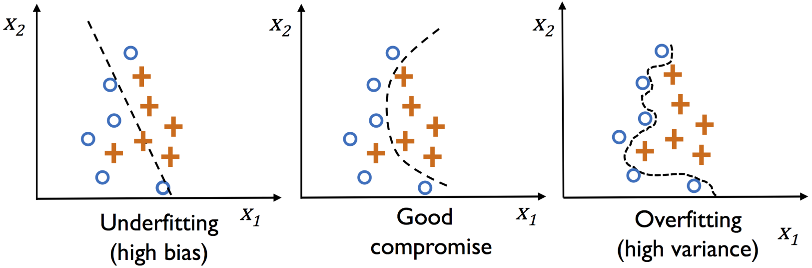
Training a logistic regression model with scikit-learn

- See details in `ch3.ipynb`

Overfitting and underfitting

- **Overfitting** is a common problem in machine learning, where a model performs well on training data but does not generalize well to unseen data (test data)
- If a model suffers from overfitting, we also say that the model has a high variance, which can be caused by having too many parameters that lead to a model that is too complex given the underlying data
- Similarly, our model can also suffer from **underfitting** (high bias), which means that our model is not complex enough to capture the pattern in the training data well and therefore also suffers from low performance on unseen data
- Figure 4 illustrates overfitting and underfitting (see next page)

Figure 4



x_2
x_1
Underfitting
(high bias)
x_2
x_1
Good
compromise
x_2
x_1
Overfitting
(high variance)

Regularization

- One way of finding a good bias-variance tradeoff is to tune the complexity of the model via regularization
- Regularization is a very useful method to handle collinearity (high correlation among features), filter out noise from data, and eventually prevent overfitting
- The idea behind regularization is to introduce additional information (bias) to penalize extreme parameter (weight) values

Regularization

- The most common form of regularization is so-called L2 regularization (sometimes also called L2 shrinkage or weight decay), which can be written as follows:

$$\frac{\lambda}{2}\|\mathbf{w}\|^2 = \frac{\lambda}{2}\sum_{j=1}^{m} w_j^2 \tag{1}$$

  where λ is the so-called regularization parameter

- The cost function for logistic regression can be regularized by adding a simple regularization term, which will shrink the weights during model training:

$$J(\mathbf{w}) = -\sum_{i=1}^{n}\left[y^{(i)}\log\left(\phi(z^{(i)})\right) - \left(1 - y^{(i)}\right)\log\left(1 - \phi(z^{(i)})\right)\right] + \frac{\lambda}{2}\|\mathbf{w}\|^2 \tag{2}$$

Regularization

- Via the regularization parameter λ, we can then control how well we fit the training data while keeping the weights small
  - by increasing the value of λ, we increase the regularization strength
- The parameter `C` that is implemented for the `LogisticRegression` class in scikit-learn is the inverse of the regularization parameter λ
  - by decreasing the value of `C`, we increase the regularization strength
- See details in `ch3.ipynb`