Lecture 6: Logistic Regression

Dr. Huiping Cao

Outline

- Theory behind the logistic regression model
- Learning the model weights via the logistic loss function
- Training a logistic regression model with scikit-learn
- Tackling overfitting via regularization

Introduction

- Logistic regression: simple, yet powerful, algorithm
- Note that, despite its name, logistic regression is a model for classification, not regression.
- It is one of the most widely used algorithms for classification in industry.
- The logistic regression model in this lecture is a linear model for binary classification.
    - Logistic regression can be readily generalized to multiclass settings, which is known as multinomial logistic regression, or softmax regression. (details see the textbook)

Concepts - odds

- Logistic regression is a probabilistic model for binary classification.
- Concept of odds: the odds in favor of a particular event.
  - The odds can be written as $\frac{p}{1-p}$, where p is the probability for the positive event.
  - The positive event refers to the event that we want to predict.
    - For example, the probability that a patient has a certain disease given certain symptoms.
    - We can think of the positive event as class label y = 1 and the symptoms as features x.
    - $p = p(y = 1|x)$, the conditional probability that a particular example belongs to a certain class 1 given its features, x.

Concepts – logit function

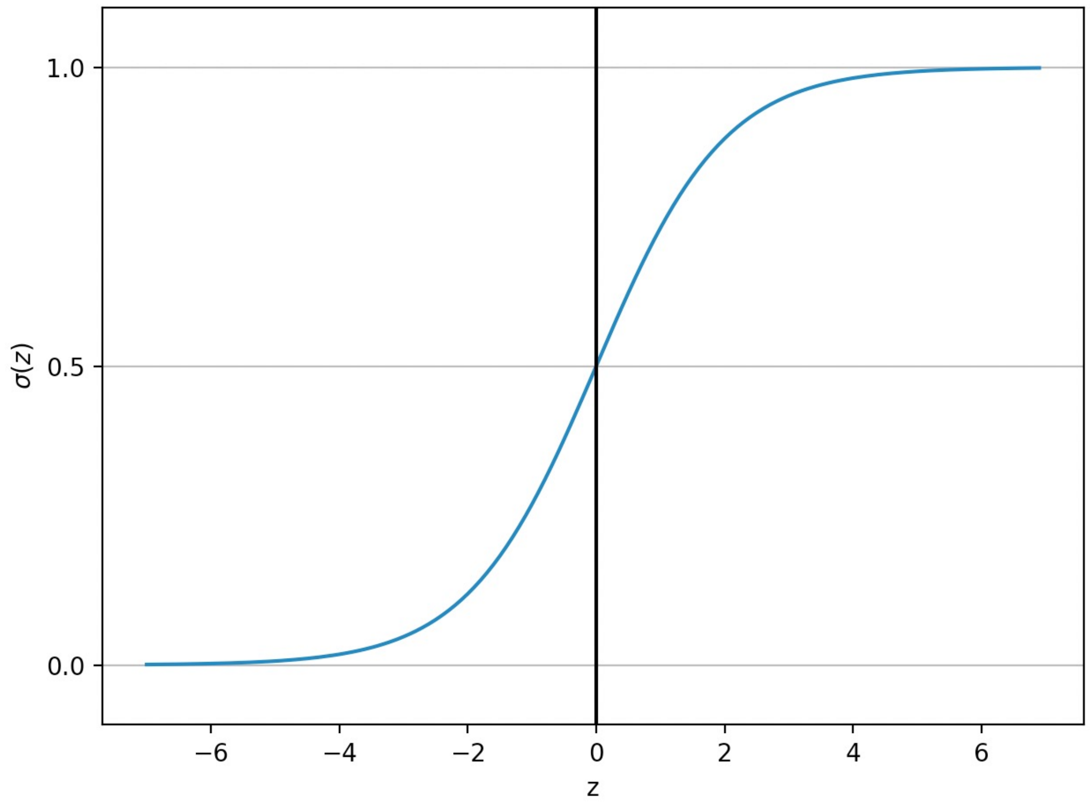
- Logit function is the logarithm of the odds (log-odds)
  - $logit(p) = log \frac{p}{(1-p)}$ where *log* refers to the natural logarithm
- Logit function takes input values in the range 0 to 1 (the probability) and transforms them into values over the entire real-number range.

What do we learn from the model

- Learn the probability p the class-membership probability of an example given its features.
- We assume that the logit function equals to the output of a linear function, i.e., $logit(p) = \boldsymbol{w}^T x + b$
- How can we learn p?
  - Logit function maps the probability to a real-number range
  - The inverse of the logit function, called logistic sigmoid function (or sigmoid function), maps the real-number range back to a [0, 1] range for the probability p.
  - Let z = $\boldsymbol{w}^T x + b$, $\sigma(z) = \frac{1}{1+e^{-z}}$

Sigmoid function

- z = $\boldsymbol{w}^T x + b$, $\sigma(z) = \frac{1}{1+e^{-z}}$
- Sigmoid function (S shape)
- $\sigma(z)$ approaches 1 if z goes toward infinity ($z \to \infty$) since e^{-z} becomes very small for large values of z.
- $\sigma(z)$ goes toward 0 for $z \to -\infty$ as a result of an increasingly large denominator.
- This sigmoid function takes real-number values as input and transforms them into values in the range [0, 1] with an intercept at $\sigma(0)$ = 0.5



Output of the sigmoid function

- The output of the sigmoid function is interpreted as the probability of a particular example belonging to class 1.
- I.e.., $\sigma(z)=p(y=1|\boldsymbol{x}; \mathbf{w}, \mathrm{b})$, given the features, x, and parameterized by the weights and bias, **w** and b.
- Example: if we compute $\sigma(z)$ = 0.8 for a particular example, it means that the chance that this example is a positive event is 80%, while the probability that this instance belonging to a negative event is 20%
  - $p(y = 0|x; \mathbf{w}, b) = 1 – p(y = 1|x; \mathbf{w}, b) = 0.2$

Prediction

- The predicted probability can then simply be converted into a binary outcome via a threshold function

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

- This is equivalent to

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

- Advantage: we get not only the predicted class label, but also the estimation of the class-membership probability

Outline

- Theory behind logistic regression
- Learning the model weights via the logistic loss function
- Training a logistic regression model with scikit-learn
- Tackling overfitting via regularization

Learning the logistic regression model

- How can we learn/fit the parameters of the model, the weights and bias unit, **w** and b?
- Goal: minimize the error (mean squared error loss function)

$$L(\boldsymbol{w}, b|x) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{2}(\sigma\left(z^{(i)}\right) - y^{(i)})^2$$

- We need to derive/define the loss function to achieve this goal

Likelihood

- We define **the likelihood, L**, that we want to maximize when we build a logistic regression model
  - Assuming that the individual examples in our dataset are independent of one another.
  - The formula is as follows

$$\mathcal{L}(\boldsymbol{w}, b|x) = p(y|x;\boldsymbol{w}, b)$$

$$= \prod_{i=1}^{n} p\left(y^{(i)}|x^{(i)};\boldsymbol{w}, b\right) = \prod_{i=1}^{n}\left(\left(\sigma\left(z^{(i)}\right)\right)^{y^{(i)}}\left(1-\sigma\left(z^{(i)}\right)\right)^{1-y^{(i)}}\right)$$

Log-likelihood and loss function

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

$$\log(\mathcal{L}(\boldsymbol{w}, b|x)) = \sum_{i=1}^{n}\left[y^{(i)}\log(\sigma(z^{(i)})) + (1 - y^{(i)})\log(1 - \sigma(z^{(i)}))\right]$$

- Rewrite the log-likelihood as a loss function, L, that can be minimized using gradient descent.

$$L(\boldsymbol{w}, b|x) = -\log(\mathcal{L}(\boldsymbol{w}, b|x)) = \sum_{i=1}^{n}\left[-y^{(i)}\log(\sigma(z^{(i)})) - (1 - y^{(i)})\log(1 - \sigma(z^{(i)}))\right]$$

Loss function – more details

- The loss for each training sample

$$L(\sigma(z), y; \boldsymbol{w}, b) = -ylog\big(\sigma(z)\big) - (1 - y)\log(1 - \sigma(z))$$

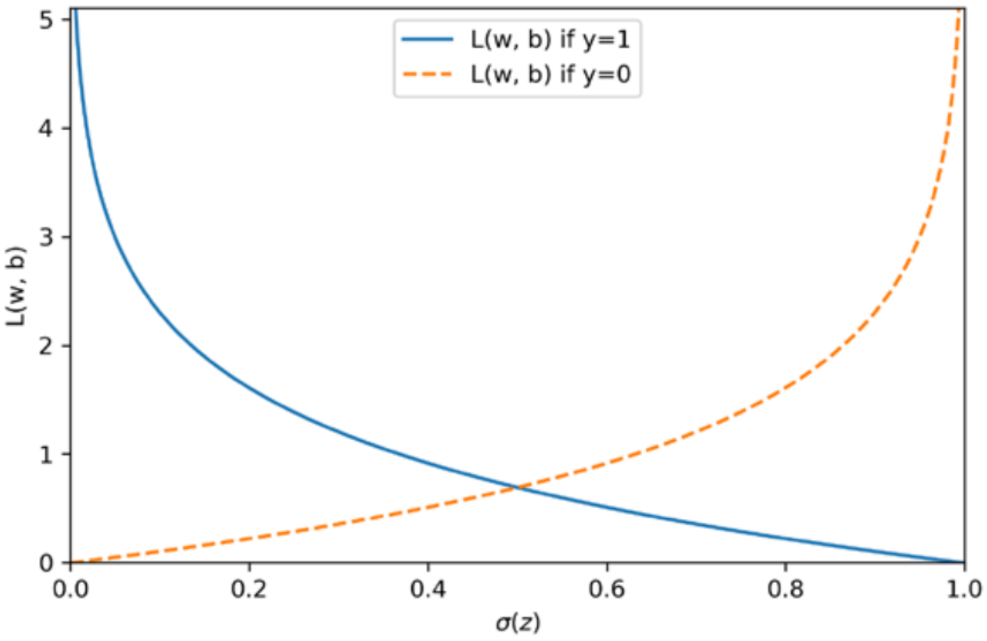
- This equation is equivalent to

$$L(\sigma(z), y; \boldsymbol{w}, b) = \begin{cases} -\log\big(\sigma(z)\big). & if\ y = 1 \\ -\log\big(1 - \sigma(z)\big) & if\ y = 0 \end{cases}$$

Loss function – more details

- $L(\sigma(z), y; \boldsymbol{w}, b) = \begin{cases} -\log(\sigma(z)). & if\ y = 1 \\ -\log(1 - \sigma(z)) & if\ y = 0 \end{cases}$

- Figure
    - x axis is in the range of 0 to 1, representing the probability of predicting an instance to be class label 1
    - y axis is logistic loss



Outline

- Theory behind logistic regression
- Learning the model weights via the logistic loss function
- Training a logistic regression model with scikit-learn
- Tackling overfitting via regularization

Logistic regression function in Scikit-learn

- Scikit-learn's implementation of logistic regression supports both binary and multiclass settings off the shelf.
  - In recent versions of scikit-learn, the technique used for multiclass classification, multinomial, or One versus Rest (OvR), is chosen automatically.
- Use the LogisticRegression function in the sklearn.linear_model module

```
>>> from sklearn.linear_model import LogisticRegression
>>> lr = LogisticRegression(C=100.0, solver='lbfgs',
...                         multi_class='ovr')
>>> lr.fit(X_train_std, y_train)
```

Define and train logistic repression model

- Hyper parameters
  - **multi_class** can be set to multi_class='multinomial'. The multinomial setting is now the default choice in scikit-learn's LogisticRegression class and recommended in practice for mutually exclusive classes
  - **Solver**: one algorithm for convex optimization. For the logistic regression classification algorithm, we need to minimize convex loss functions, which is the logistic regression loss.
  - **C**: a parameter related to control overfitting; discussed later

```
>>> from sklearn.linear_model import LogisticRegression
>>> lr = LogisticRegression(C=100.0, solver='lbfgs',
...                         multi_class='ovr')
>>> lr.fit(X_train_std, y_train)
```

Prediction using logistic regression model

- The logistic regression model can be used to predict class membership probabilities using the **predict_proba** method.

```
>>> lr.predict_proba(X_test_std[:3, :])
```

- Results
  - The i-th row corresponds to the class membership probabilities of the i-th flower
  - The column-wise sum in each row is 1, as expected.

```
array([[3.81527885e-09, 1.44792866e-01, 8.55207131e-01],
       [8.34020679e-01, 1.65979321e-01, 3.25737138e-13],
       [8.48831425e-01, 1.51168575e-01, 2.62277619e-14]])
```

Prediction using logistic regression model

- To directly predict class labels, the scikit-learn **predict** method can be used.

```
>>> lr.predict(X_test_std[:3, :])
array([2, 0, 0])
```

- Note: if you want to predict the class label of a single flower example.
  - scikit-learn expects a two-dimensional array as data input thus, we have to convert a single row slice into such a format first.
  - One way to convert a single row entry into a two-dimensional data array is to use NumPy's reshape method to add a new dimension].

```
>>> lr.predict(X_test_std[0, :].reshape(1, -1))
array([2])
```

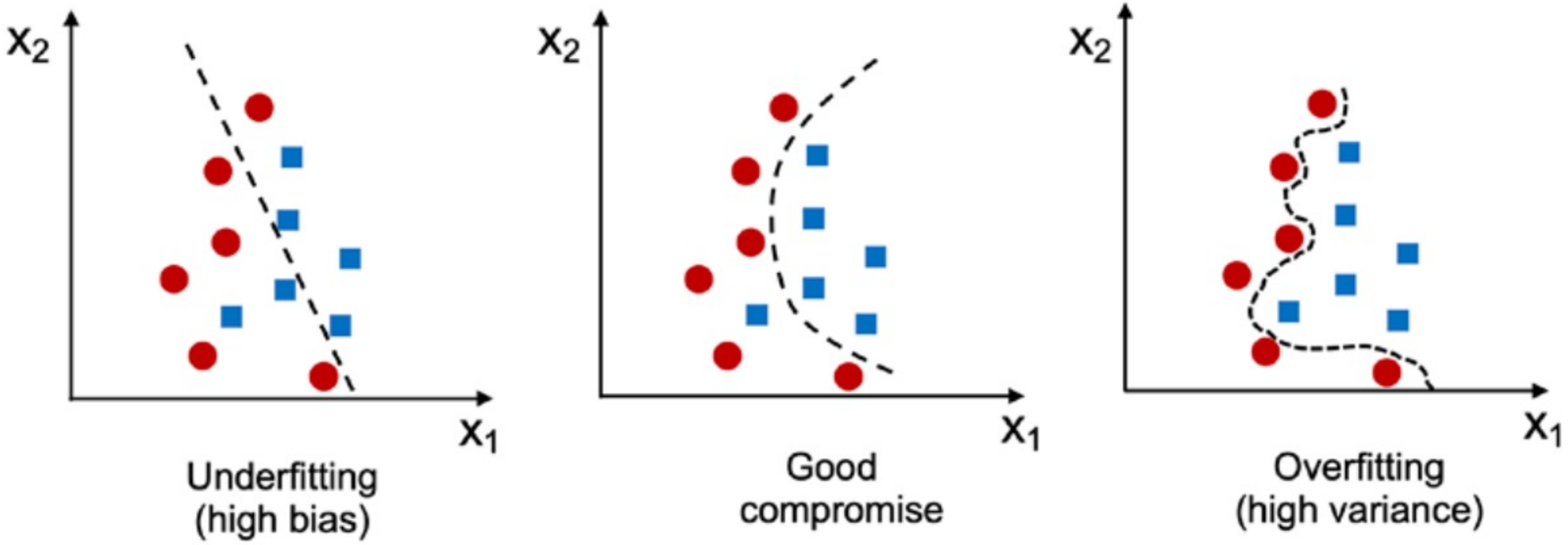
Outline

- Theory behind logistic regression
- Learning the model weights via the logistic loss function
- Training a logistic regression model with scikit-learn
- Tackling overfitting via regularization

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.
  - Possible reason for overfitting: models have too many parameters, leading to a model that is too complex given the underlying data.
- Underfitting (high bias) means that a model is not complex enough to capture the pattern in the training data well and therefore also suffers from low performance on unseen data.

Examples



- Examples of underfitted, well-fitted, and overfitted models
- One way of finding a good bias-variance tradeoff is to tune the complexity of the model via regularization.

Regularization

- Regularization is a very useful method for handling (i) collinearity (high correlation among features), (ii) filtering out noise from data, and (iii) eventually preventing overfitting.
- Introduce additional information to penalize extreme parameter (weight) values.
- The most common form of regularization is so-called L2 regularization (sometimes also called L2 shrinkage or weight decay).
  - λ is the so-called regularization parameter

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

Regularization and feature scaling

- Regularization is another reason why feature scaling such as standardization is important.
- For regularization to work properly, we need to ensure that all our features are on comparable scales.

Regularized loss function

- The loss function for logistic regression can be regularized by adding a simple regularization term.

$$L(\boldsymbol{w}, b|x) = \frac{1}{n}\sum_{i=1}^{n}\left[-y^{(i)}\log\left(\sigma\left(z^{(i)}\right)\right) - \left(1 - y^{(i)}\right)\log\left(1 - \sigma\left(z^{(i)}\right)\right)\right]$$

$$\Downarrow$$

$$L(\boldsymbol{w}, b|x) = \frac{1}{n}\sum_{i=1}^{n}\left[-y^{(i)}\log\left(\sigma\left(z^{(i)}\right)\right) - \left(1 - y^{(i)}\right)\log\left(1 - \sigma\left(z^{(i)}\right)\right)\right] + \frac{\lambda}{2n}||\boldsymbol{w}||^2$$

Parameter λ

- Loss function

$$L(\boldsymbol{w}, b|x) = \frac{1}{n}\sum_{i=1}^{n}[-y^{(i)}\log(\sigma(z^{(i)})) - (1 - y^{(i)})\log(1 - \sigma(z^{(i)}))] + \frac{\lambda}{2n}||\boldsymbol{w}||^2$$

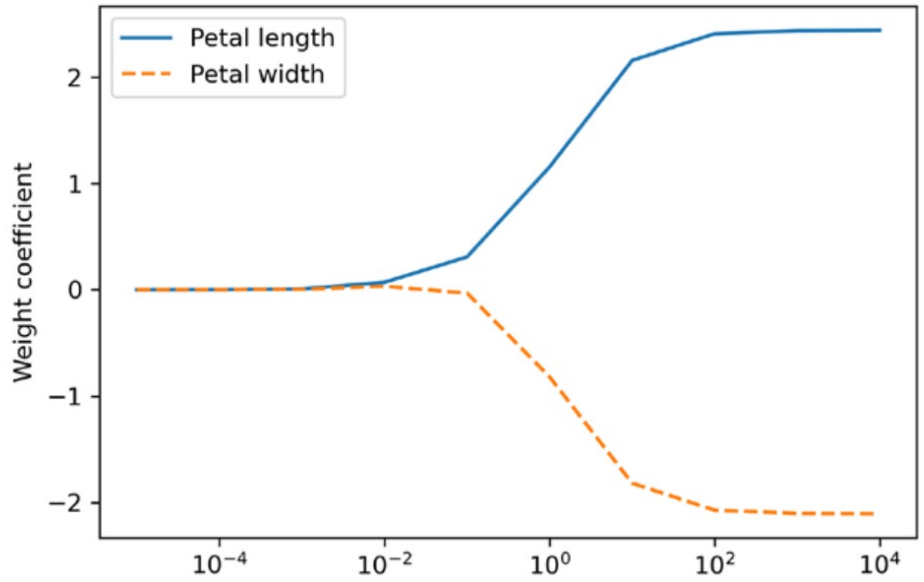
- Partial derivative

$$\frac{\partial L(\boldsymbol{w}, b)}{\partial w_j} = \left(\frac{1}{n}\sum_{i=1}^{n}(\sigma(\boldsymbol{w}^T\boldsymbol{x}^{(i)}) - y^{(i)})x_j{}^{(i)}\right) + \frac{\lambda}{n}w_j$$

- By increasing the value of λ, we increase the regularization strength.
- Bias unit is generally not regularized.

Inverse-regularization parameter C

- The inverse-regularization parameter C is inversely proportional to the regularization parameter λ.
- Decreasing the value of C, we increase the regularization strength.
- We can examine the effect of C over the weight coefficients by fitting multiple logistic regression models with different values for C.



The impact of C on L2 regularized model results

References

- Sebastian Raschka, Yuxi Liu, Vahid Mirjalili: Machine Learning with PyTorch and scikit-learn. 3rd Edition. ISBN 978-1-80181-931-2. Publisher: Packt Publishing Ltd. **Chapter 3**