6. LOGISTIC REGRESSION

Logistic regression is an example of a classifier which is used to estimate the probability that an instance belongs to a particular class.

It was proposed in 1940s by various authors.

- Consider the binary classification problem with data

$$D = \{(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})\}$$

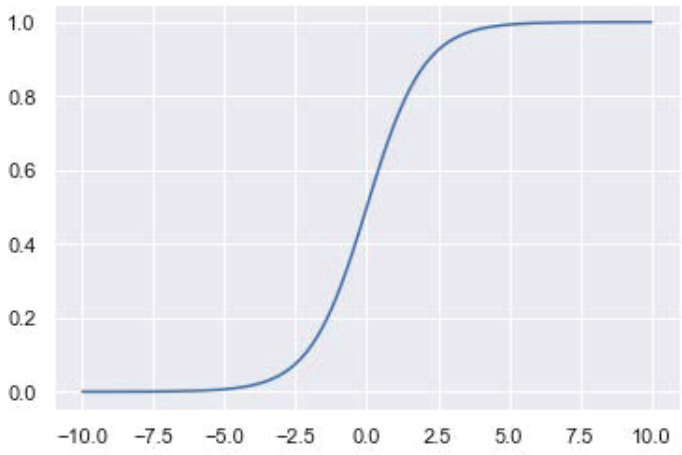
where $x^{(i)} \in \mathbb{R}^d$ and $y^{(i)} \in \{+1, -1\}$. Instead of making predictions $\hat{y}^{(i)} \in \{+1, -1\}$, logistic regression generates real-valued outputs in the interval $(0, 1)$. A linear logistic classifier has the form

$$\boxed{h(x; w, b) = \sigma(w^T x + b)}$$

where $w \in \mathbb{R}^d$ and $b \in \mathbb{R}$ are parameters and

$$\sigma(z) = \frac{e^z}{e^z + 1} \quad \text{or} \quad \boxed{\sigma(z) = \frac{1}{1 + e^{-z}}}$$

is the sigmoid function. The graph of the sigmoid function is



Homework Questions:

1. Why is output of σ in the interval $(0, 1)$?
2. Why the output of σ cannot equal 0 or 1?
3. For what value of z is $\sigma(z) = 0.5$?
4. Show that $\sigma'(z) = \sigma(z) \cdot (1 - \sigma(z))$.

- Given that the output of the logistic regression is a value in $(0, 1)$, if we need to make a prediction for a label in $\{+1, -1\}$, by default
  - we classify $x^{(i)}$ as an positive example if $\sigma(w^T x^{(i)} + b) > 0.5$

– we classify $x^{(i)}$ as a negative example if $\sigma(w^T x^{(i)} + b) < 0.5$

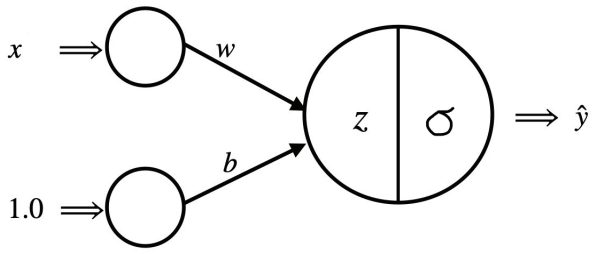
Notice that this default decision boundary is given by the hyperplane $w^T x + b = 0$.

The value of 0.5 is called a *prediction threshold*. Depending on the actual application, we might want to pick a different prediction threshold. For example, if the consequence of predicting -1 when the actual value is $+1$ is much worse than the consequence of predicting $+1$ when the actual answer is -1, we might set the prediction threshold to be a value less than 0.5 (consider the problem of classifying patients where positive class contains patients who have diabetes).

- Note that we can view the logistic regression as a single neuron with preactivation

$$z = w^T x + b$$

and the activation function being the sigmoid function. In case $x \in \mathbb{R}$ we have a diagram



- Note that logistic regression outputs the probability for a given value of x by the formula

$$\hat{y} = \sigma(w^T x + b) = \frac{1}{1 + e^{-(w^T x + b)}} = p.$$

Rewriting this formula we get

$$\log \frac{p}{1-p} = w^T x + b$$

where log stands for the natural logarithm. The quantity on the left is called *log-odds* or *logit*. Therefore, logistic regression model has a logit that is a linear function of x.

- To determine the parameters w and b, we will define the *loss function*. Recall that predicted outputs are $\hat{y}^{(i)} \in (0, 1)$, while the true target values are $y^{(i)} \in \{+1, -1\}$. Intuitively, we would like to have low loss if we assign a low probability to the incorrect class.

To simplify description, we convert the labels in the training set to be $y^{(i)} \in \{0, +1\}$, so we can interpret them as probabilities of being a member of the class of interest. To find the parameters w and b, we use a method called *maximum likelihood* (recall the probabilistic interpretation for linear regression).

We would like to pick w and b to <u>maximize</u> the probability assigned by the logistic regression to the correct label

$$P(y^{(i)}|x^{(i)}) = \begin{cases} \hat{y}^{(i)} & \text{if} \quad y^{(i)} = 1 \\ 1 - \hat{y}^{(i)} & \text{if} \quad y^{(i)} = 0 \end{cases}$$

This can be cleverly rewritten as

$$P(y^{(i)}|x^{(i)}) = \left(\hat{y}^{(i)}\right)^{y^{(i)}} \cdot \left(1-\hat{y}^{(i)}\right)^{1-y^{(i)}}$$

Assuming that our data instances are independent, that probability over the entire training data set is

$$\prod_{i=1}^{n} P(y^{(i)}|x^{(i)}) = \prod_{i=1}^{n} \left(\hat{y}^{(i)}\right)^{y^{(i)}} \cdot \left(1-\hat{y}^{(i)}\right)^{1-y^{(i)}}$$

where

$$\hat{y}^{(i)} = \sigma(w^T x^{(i)} + b) = \frac{1}{1+e^{-(w^T x^{(i)}+b)}}$$

Recall that our goal is to maximize the above product in terms of w and b. Since the logarithmic function with base higher than 1 is monotonically increasing, the parameters w and b that maximize the above product are the same parameters that maximize the log of that product, which is

$$\sum_{i=1}^{n} \left(y^{(i)} \log \hat{y}^{(i)} + (1-y^{(i)}) \log(1-\hat{y}^{(i)})\right)$$

Lastly, we covert the maximization problem to a minimization problem by taking the negative of the above expression. In other words, we want to find parameters w and b that minimize

$$\boxed{L(w,b;D) = -\frac{1}{n}\sum_{i=1}^{n} \left(y^{(i)} \log \hat{y}^{(i)} + (1-y^{(i)}) \log(1-\hat{y}^{(i)})\right)}$$

This loss function L is called *log loss* (more precisely, *negative log-likelihood loss*) or *cross entropy loss* (more precisely, *binary cross entropy loss*).

Remarks:

- You can use any base higher than 1 for the logarithm.
- Let us find the binary cross entropy loss for the 5th data instance for which $y^{(5)} = 0$ and $\hat{y}^{(5)} = 0.91$

$$\begin{aligned} \text{loss} &= -\left(y^{(5)} \log \hat{y}^{(5)} + (1-y^{(5)}) \log(1-\hat{y}^{(5)})\right) \\ &= -\left(0 \cdot \log(0.91) + 1 \cdot \log(0.09)\right) \\ &= 2.41 \end{aligned}$$

- Let us find the binary cross entropy loss for the 7th data instance for which $y^{(7)} = 0$ and $\hat{y}^{(7)} = 0.11$

$$\begin{aligned} \text{loss} &= -\left(y^{(7)} \log \hat{y}^{(7)} + (1-y^{(7)}) \log(1-\hat{y}^{(7)})\right) \\ &= -\left(0 \cdot \log(0.11) + 1 \cdot \log(0.89)\right) \\ &= 0.12 \end{aligned}$$

- There is no known closed-form analytical solution for parameters w and b that minimize the above loss function. However, since the loss function is convex, the gradient descent method is guaranteed to find the global minimum, provided that the learning rate is not too large and we use sufficiently many iterations.

  For simplicity, consider the one-dimensional case where $x^{(i)} \in \mathbb{R}$ and

  $$\hat{y}^{(i)} = \sigma(wx^{(i)} + b)$$

  with $w, b \in \mathbb{R}$. We use that $\sigma'(z) = \sigma(z) \cdot (1 - \sigma(z))$ and the chain rule to find

  $$\frac{\partial L}{\partial w} = -\frac{1}{n} \sum_{i=1}^{n} \left(y^{(i)} \frac{1}{\hat{y}^{(i)}} \hat{y}^{(i)}(1 - \hat{y}^{(i)}) x^{(i)} + (1 - y^{(i)}) \frac{1}{1 - \hat{y}^{(i)}} (-\hat{y}^{(i)})(1 - \hat{y}^{(i)}) x^{(i)} \right)$$

  $$\boxed{\frac{\partial L}{\partial w} = \frac{1}{n} \sum_{i=1}^{n} \left(\hat{y}^{(i)} - y^{(i)} \right) x^{(i)}}$$

  Similarly,

  $$\boxed{\frac{\partial L}{\partial b} = \frac{1}{n} \sum_{i=1}^{n} \left(\hat{y}^{(i)} - y^{(i)} \right)}$$

  Remarks:

  - Look up these formulas for derivatives and the updating rules when L was the mean squared error in linear regression in §5. We can reuse our code for linear regression and train logistic regression similarly as well as use batch gradient descent, stochastic gradient descent, or mini-batch gradient descent. It is not a coincidence that these formulas are very similar and, in fact, both linear and logistic regression are special cases of Generalized Linear Models (GLM).
  - Also, look up the update formulas for perceptron in §3. Even though the formulas are very similar, the perceptron is a very different algorithm and it is difficult to endow its predictions with a meaningful probabilistic interpretation and derive the perceptron as a maximum likelihood estimate algorithm.

- **Categorical inputs and dummy variables**

  If the inputs are categorical variables, they need to be encoded and most of the time we use dummy variables and one-hot encoding.

  For example, if we have a salary input with values "low", "medium", "high", we can encode it using "1", "2", "3". However, this implies that

  $$\text{"medium"} = 2 \cdot \text{"low"} \quad \text{and} \quad \text{"high"} = 3 \cdot \text{"low"}$$

  To avoid this issue, we use one-hot encoding by adding three more binary input variables.

salary	**salary_low**	**salary_medium**	**salary_high**
medium	0	1	0
low	1	0	0
high	0	0	1

We can drop column "salary". Also note that the remaining three columns are not independent and to avoid this we can drop one of the three columns.

- **Multiclass Logistic Regression**

  We described above the linear logistic model for binary classification, however binary classifiers can be extended for multiclass classification problems.

  Two most popular approaches to extending binary classifiers from binary to K-class classification with $K > 2$ are as follows.

  - *One Vs. One Classification*

    In this approach we construct $\binom{K}{2}$ binary classifiers, each of which compares a pair of classes. We classify a test observation using the majority voting among these $\binom{K}{2}$ classifiers.

  - *One Vs. All (or One Vs. Rest) Classification*

    We fit K binary classifiers, each time comparing one of the classes (coded as a positive class) to the remaining group of $K-1$ classes (coded as a negative class). Let w_k and b_k be parameters from this classification, $k \in \{1, \ldots, K\}$. If x^* is a test data instance, we assign it to the class for which $(w_k)^T x^* + b_k$ is the largest as this amounts to the high level of confidence that x^* belonds to that class.

Evaluation methods and metrics for classifiers

- The **confusion matrix** is a way to summarize how the model performed. It is a square matrix whose size is the same as the number of different output labels. In `sklearn`, the rows correspond to the observed or the actual labels and the columns correspond to the labels predicted by the model. The diagonal elements in the matrix contain the number of data instances that were correctly classified by the model, while the other entries contain the number of data instances that were misclassified by the model.

  For binary classification problems, the confusion matrix is given by

	predicted negative	predicted positive
actual negative	true negative (TN)	false positive (FP)
actual positive	false negative (FN)	true positive (TP)

  Remarks:

  - Note that FP corresponds to Type I statistical error and FN corresponds to Type II statistical error, and we always try to reduce these two errors.
  - The classes in the sklearn confusion matrix are ordered alphabetically.

- We use the confusion matrix to define many different measures.

  - **Accuracy** is the percentage of correct predictions.

$$\boxed{\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}}$$

    It is used when both classes are equally important. For example, classifying images as images of dogs and cats.

  - **Precision** is the percentage of the positive predictions which are correct.

$$\boxed{\text{Precision} = \frac{\text{TP}}{\text{TP + FP}}}$$

    Precision is useful when the costs of FP is high and we want to reduce the number of FP.

    In sklearn classification report, precision is reported for each predicted class as the percentage of correct predictions. For binary classification,

$$\text{Precision (positive class)} = \frac{\text{TP}}{\text{TP + FP}}$$

$$\text{Precision (negative class)} = \frac{\text{TN}}{\text{TN + FN}}$$

  - **Recall** is the percentage of positive observations that were correctly classified.

$$\boxed{\text{Recall} = \text{Sensitivity} = \text{True Positive Rate (TPR)} = \frac{\text{TP}}{\text{TP + FN}}}$$

    Recall is useful when the cost of FN is high and we want to reduce the number of FN.

In sklearn classification report, recall is reported for each observed class as the percentage of correct predictions.

$$\text{Recall (positive class)} = \text{Sensitivity} = \text{True Positive Rate (TPR)} = \frac{\text{TP}}{\text{TP + FN}}$$

$$\text{Recall (negative class)} = \text{Specificity} = \text{True Negative Rate (TNR)} = \frac{\text{TN}}{\text{TN + FP}}$$

Depending on whether correctly identifying positives or negatives is more important, we choose the model with the highest sensitivity or specificity. For example, in testing whether a patient has cancer or not, we want to reduce the number of FN (or, equivalently, increase sensitivity). It is a disaster if a person has cancer and the test was negative since the patient will not be offered the medical treatment. On the other hand, we do not care as much about specificity since if the person is FP, meaning the person does not have cancer, but was classified as positive, the person would have to go to additional tests, and the potential damage is not as bad.

- F_1**-score** is defined, for each class, as the harmonic average of the recall and the precision

$$\boxed{F_1 = \frac{2 \cdot \text{Recall} \cdot \text{Precision}}{\text{Recall} + \text{Precision}}}$$

Harmonic average is just another way to calculate an "average" of values, generally described as more suitable for ratios (such as precision and recall) than the traditional arithmetic average. For two numbers x_1 and x_2, the arithmetic average is defined as

$$\text{arithmetic average} = \frac{x_1 + x_2}{2}$$

while the harmonic average is defined as

$$\text{harmonic average} = \frac{2}{\frac{1}{x_1} + \frac{1}{x_2}} = \frac{2x_1x_2}{x_1 + x_2}$$

The idea behind the F_1-score is to provide a measure that takes precision and recall in a balanced way, requiring both to have a higher value for the F_1-score to be high. For example, a precision of 0.01 and recall of 1.0 give

$$\text{arithmetic average} = \frac{0.01 + 1.0}{2} = 0.505$$

$$\text{harmonic average} = \frac{2 \cdot 0.01 \cdot 1.0}{0.01 + 1.0} = 0.02$$

We note that harmonic average (which is really the F_1-score) is much more sensitive to one of the two inputs having a low value (0.01 here). Very small precision or very small recall will result in lower overall F_1 score.

- **Remarks on evaluation metrics**
  - Accuracy is a good measure if we have balanced datasets and are interested in all types of outputs equally. We usually start with the accuracy score and dig deeper from there as needed.

- Precision is great to focus on if we want to minimize FP. For example, if we build a spam email classifier, we do not want to miss any important, non-spam emails. In such case, we want to maximize precision.
- Recall is very important in domains such as medical (for example, identifying cancer), where we want to minimize the chance of missing positive cases or, in other words, predicting FN.
- Accuracy is used when TP and TN are more important while F_1-score is used when the FN and FP are crucial.
- F_1-score combines precision and recall, and works also for cases where the datasets are imbalanced.

Python code: Lecture_6_Logistic_Regression.ipynb

Homework 3:

- Answer Homework Questions on page 1.
- Use `HR.csv` data set and consider column "left" to be the target variable with "1" meaning the person left the company and "0" meaning that the person did not leave the company.
  - Investigate using various graphs/charts how given features affect this target variable.
  - Choose several features and build the `sklearn` logistic regression model predicting the target variable "left".
  - Discuss the model performance (the confusion matrix and the classification report) on the test set.

References and Reading Material:

[1] HR.csv data set:
https://www.kaggle.com/datasets/kmldas/hr-employee-data-descriptive-analytics

[2] *Hands on Machine Learning with Scitkit-Learn, Keras & TensorFlow*, Geron, pages 142-147

[3] Andrew Ng's notes, pages 16-19 (probabilistic approach – optional, pages 22-30)

[4] *An Introduction to Statistical Learning*, James et. al. (Sections 4.1-4.3, pages 127-138)

[5] MIT notes, Chapter 5, pages 29-32