Logistic Regression

Department of Computing and Mathematical Sciences
University of Greenwich, London

Feb 19, 2021

Outline

Logistic Regression

- Simple yet powerful algorithm.
- Probably the most widely used ML algorithm in the industry.
- Often used as a baseline algorithm against which more *advanced* algorithms' performances are compared.
- Not to be confused with Linear Regression since it may differ in terms of prediction task (classification – discrete values rather than regression – continuous values). Also, activation and cost/loss functions are different [**important difference!**].
- However, logistic regression is **regressed** on the probability of the categorical (e.g., binary) outcome. More on this later.

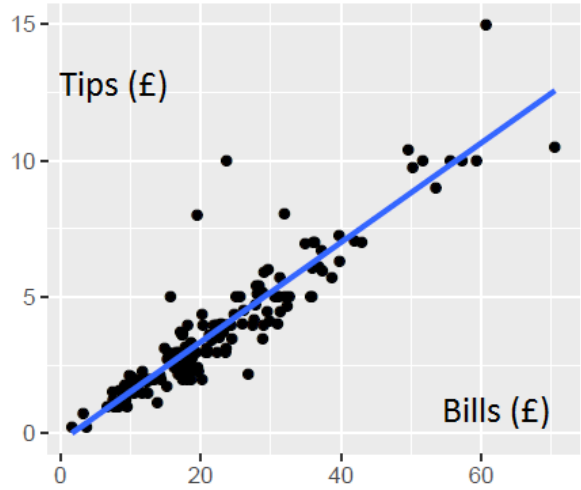


Figure: predict prob. of a huge tip given bills

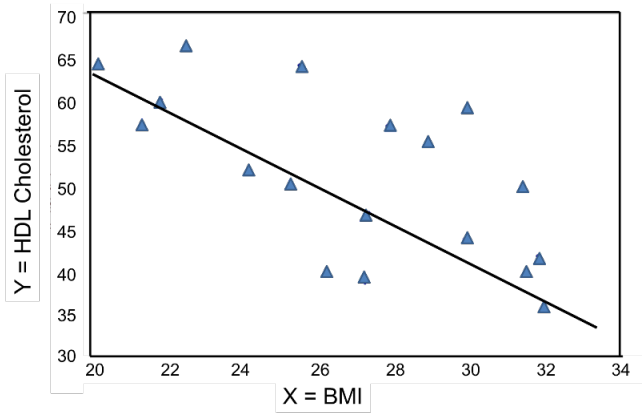
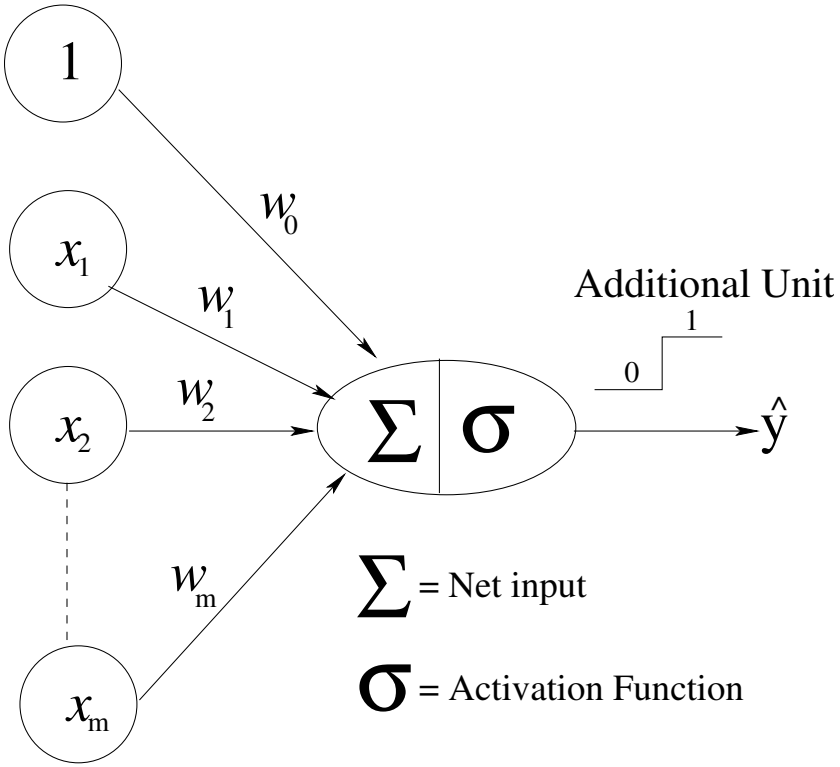


Figure: predict prob. of high cholesterol given BMI

Logistic Regression

Suppose, a sample, X contains n records - each with m features, and a target variable, y. A **supervised** classification task's goal is to predict the target variable (discrete) given the features.



- Linear Regression:
  - Activation function (σ) is the linear function, i.e., $\sigma(z) = z$.
  - Output, $\hat{y} \in \mathbb{R}$ (real numbers).
- Logistic Regression:
  - Activation function (σ) is the sigmoid function, i.e., $\sigma(z) = \frac{1}{1+e^{-z}}$.
  - Additional unit:

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

  - Output, $\hat{y} \in \{0, 1\}$ for binary classes.

(Cross Entropy) Logistic Regression

$\sigma(z)$ is the measure of probability of a class being either 1 or 0 [binary classification]. We want,

$$\sigma(z) = \begin{cases} P(y=1|x) \approx 1 \text{ if } y=1 \\ P(y=0|x) \approx 1 \text{ if } y=0 \end{cases}$$

Could we use the same loss function, $\mathcal{L}(w) = \frac{1}{2n}\sum_{i=1}^{n}(y^{[i]} - \sigma(z^{[i]}))^2$ as in Linear Regression? $\implies$ not convex!

Log-likelihood cross-entropy resolves this issue,

$$\mathcal{L}(w) = -\frac{1}{n}\sum_{i=1}^{n}\Big[(y^{[i]}\log\{\sigma(z^{[i]})\}) + (1-y^{[i]})\log\{1-\sigma(z^{[i]})\}\Big]$$

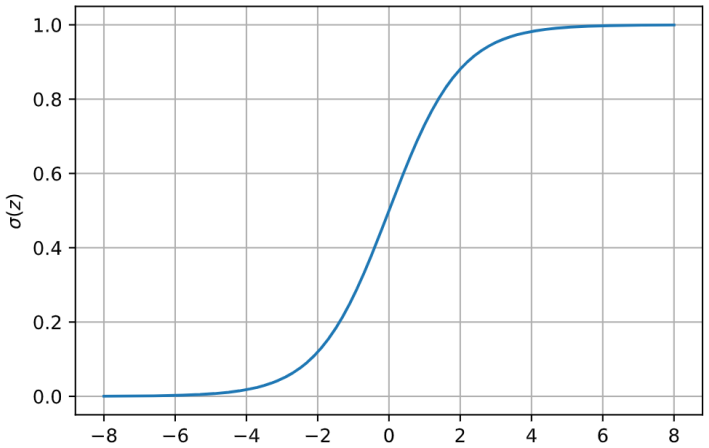


Figure: $\sigma(z)$: non-linear activation function [RM19]

(Cross Entropy) Logistic Regression

For example, if $y^{[i]} = 1$ and $\sigma(z^{[i]}) \approx 1$, then the loss for this i^{th} record, $\mathcal{L}(w) = 0$ (as it should be since correct prediction).
Refer to the $\mathcal{L}(w)$ equation in the previous slide. The 2nd term's contribution is zero. For the first term, it is also zero, $y^{[i]} \log\{\sigma(z^{[i]})\} = 1 \log 1 = 0$.
You can verify zero loss for $y^{[i]} = 0$ and $\sigma(z^{[i]}) \approx 0$ too in a similar way.

How misclassification impacts the loss function is depicted on the right hand side figure.

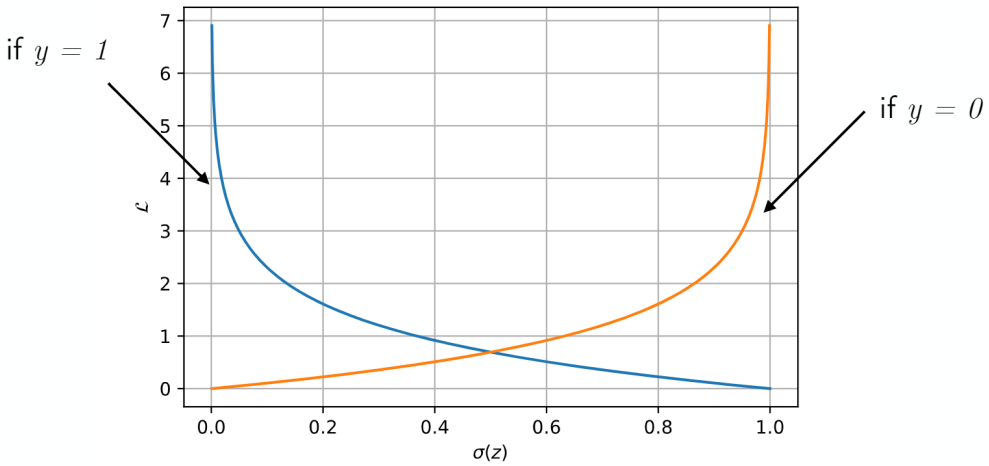


Figure: Loss function, $\mathcal{L}(w)$ [RM19]

(Cross Entropy) Logistic Regression – Gradient Descent

Initialise the weights, w

For every training epoch:

For every record i, $< X^{[i]}, y^{[i]} >$:

(a) $\hat{y}^{[i]} = \sigma(\sum_{j=0}^{m} w_j x_j)$

For each weight index, $j \in \{0, 1, \ldots\ldots m\}$:

(b) $\frac{\partial \mathcal{L}}{\partial w_j} = -(y^{[i]} - \hat{y}^{[i]}) x_j{}^{[i]}$

(c) $w_j = w_j + \eta \left(-\frac{\partial \mathcal{L}}{\partial w_j} \right)$, where $\eta =$ learning rate

End of procedure when the weights from one epoch to the next do not change or maximum number of epochs are reached.

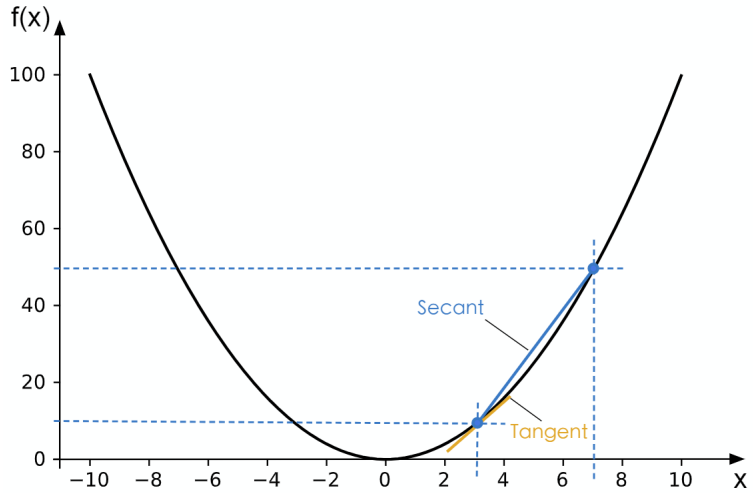


Figure: Graphical Representation of Derivatives

$$\frac{dy}{dx} = \lim_{\Delta x \to 0} \frac{f(x + \Delta x) - f(x)}{\Delta x}$$

(Cross Entropy) Logistic Regression – Gradient Descent

Scikit-Learn Logistic Regression Documentation Link here.
Research the different **'solver's** that are mentioned in the link that are available as optimisation algorithms (Look for solver parameter). Which particular solver do you think relates to 'Gradient Descent' algorithm that we discuss here?

Logistic Regression of Sklearn Library's Implementation Details Link here.

Logistic Regression – Evaluation Metrics

Measure of goodness of fit and how well the model performs.

- Accuracy $= \frac{TP+TN}{TP+FP+FN+TP}$
- Precision $= \frac{TP}{TP+FP}$, and Recall $= \frac{TP}{TP+FN}$
- F1 Score = Harmonic Mean of Precision and Recall $= \frac{2}{\frac{1}{\text{Precision}}+\frac{1}{\text{Recall}}}$
- ROC Curve: True Positive Rate (TPR/Recall/Sensitivity) vs False Positive Rate (FPR/1-Specificity) – sometimes, AUC (closer to 1.0) is a good indication of the model's performance.

Which evaluation metric is the best? Why?

	Predicted 0	Predicted 1
Actual 0	TN	FP
Actual 1	FN	TP

Figure: Confusion Matrix [source]
TP = True positive, TN = True negative
FP = False positive, FN = False negative

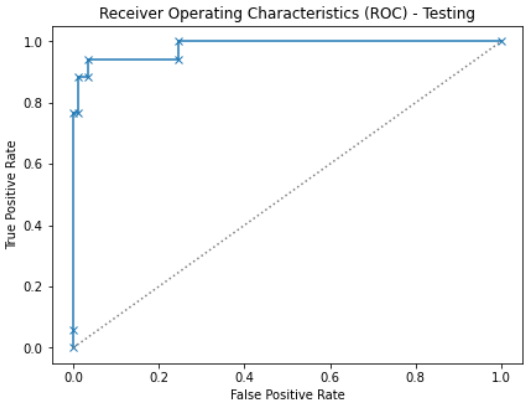


Figure: ROC Characteristics: TPR vs FPR

Logistic Regression – Regularization

- The concept is similar to what we discussed in the 'Linear Regression' slides.
- Loss function (L2 regularization),

$$\mathcal{L}(w) = -\frac{1}{n}\sum_{i=1}^{n}\Big[(y^{[i]}\log\{\sigma(z^{[i]})\}) + (1-y^{[i]})\log\{1-\sigma(z^{[i]})\}\Big] + \lambda\sum_{j=0}^{m} {w_j}^2$$

- In regularization scenarios, you may need to find the optimum λ. GridSearch can be useful - this is demonstrated inside the **Modelling (with hyper-parameter tuning)** section of 'logistic regression' notebook here.
- Remember, in scikit-learn library, the regularization parameter (C) is the inverse of λ, i.e., $C = \frac{1}{\lambda}$. This means lower value of C is interpreted as Regularization strength.

Logistic Regression – Interpretation

Inside the 'Inference Probabilities' section of the 'logistic regression' notebook here, we discussed a few insights:

- Logistic regression is regressed on Probability($Y == 1|X$). In our example, the probability of a house being expensive given the 13 features?
- The classification is just performed by introducing an additional thresholding unit, i.e., if this probability ≥ 0.5, then classify as 'Expensive' (1), otherwise, not (0).
- Logistic regression can be extended to 'multi-class' categorisation quite easily - one of your tutorial tasks.
- It gives the notion of increase/decrease in "odds" of $\Pr(Y == 1|X)$ regarding **change** in predictor variables. More details are inside the 'Inference Probabilities' section of the notebook.

Conclusion

- We discussed 'Logistic Regression' following ML landscape.
- We identified the 'learning' part regarding this algorithm. For example, clear concept of loss/cost function, and what we were trying to optimise/learn/evaluate?
- We continued the discussion of regularization for this algorithm as well.
- We interpreted Logistic Regression's application in a few ways – especially clarifying the regression part and also its applicability to linearly separable classes. It is widely used in medicine, credit scoring or even NLP tasks because of its simplicity and speed.
- It is simple yet extremely powerful and popular which forms the basis of understanding other advanced algorithms, e.g., the upcoming lectures Neural Network models.

References


Sebastian Raschka and Vahid Mirjalili.
Python Machine Learning, 3rd Ed.
Packt Publishing, Birmingham, UK, 2019.

Appendix

Interpreting odds in terms of associated weights, w_i where this algorithm is regressed on the odds. Suppose, $p = P(Y == 1|X)$, then the odd is represented as, $\frac{p}{1-p}$.

$$
\begin{aligned}
p^{\text{orig}} &= \frac{1}{1 + e^{-\sum_{j=0}^{m} w_j x_j}}, \text{ output of activation unit - Slide 4} \\
p^{\text{mod}} &= \frac{1}{1 + e^{-\sum_{j=0}^{m} w_j x_j - w_k}}, \ k^{\text{th}} \text{ feature increased by 1} \\
\text{Change in odds} &= \left(\frac{p^{\text{mod}}}{1 - p^{\text{mod}}}\right) / \left(\frac{p^{\text{orig}}}{1 - p^{\text{orig}}}\right) \\
&= e^{w_k}
\end{aligned}
$$

This motivates the interpretation of Cell 9 of the notebook here. If you consider the logit function (as in many textbooks) $\log \frac{p}{1-p}$ - this change of odds becomes only w_k (similar to linear regression).