INNOPOLIS UNIVERSITY

Machine Learning

Prof. Adil Khan

Objectives

1. A quick recap of last week
2. Gradient Descent
3. Classification Tasks
4. What is logistic regression? What is its objective function? How is it motivated?
5. Finding the optimum parameters of logistic regression
6. Classification Metrics1: Accuracy
7. Things to Know about ML-1: Class-imbalance
8. Classification Metrics2: Precision, Recall, F1-score

Reflection

- Why is predicting the number of defects in a software a regression problem?
- What do the words “simple”, “linear”, and “regression” represent in Simple Linear Regression?
- Why do we want to minimise Mean Squared Error (MSE) with respect to the parameters of our regression model?
- What is the role of a derivative in this minimisation problem?
- What are convex functions and why are we delighted that MSE is a convex function?
- What is a non-linear regression problem?
- Why is linear regression inadequate for non-linear problems?
- What is a polynomial function and how can they help us in non-linear regression problems?

Recap (1)

1. What are software bugs?
2. What are their sources?
3. What are their adverse effects?
4. How unlikely is it to create bug-free software?
5. How important is it to be able to predict defect's related information?

Predicting Number of Defects From the Point of view of ML

$$\underbrace{\#\ of\ defects\ in\ p_i}_{y_i} = f(\underbrace{features\ or\ behavior\ of\ p_i}_{\boldsymbol{x}_i})$$

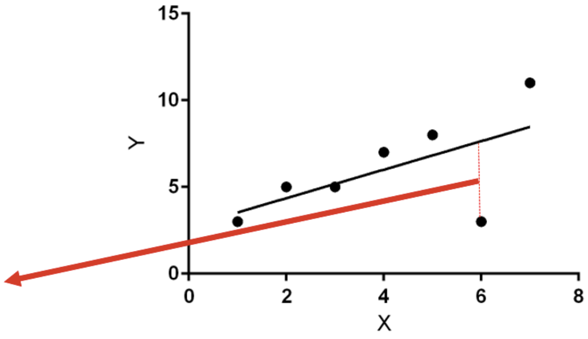
y categorical is continuous?

Thus we are dealing with a regression problem

How Do We Train Linear Regression Model?

$$f(x_i) = w_0 + w_1 x_i$$

$$e_i = y_i - f(x_i)$$



Linear Regression

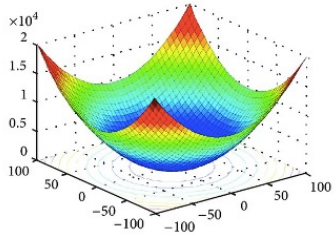
$$\boxed{y = w_0 + w_1 x_1 + w_2 x_2 + \cdots w_p x_p}$$

- The response variable is quantitative
- The relationship between response and predictors is assumed to be linear in the inputs
- Thus we are restricting ourselves to a hypothesis space of linear functions

Recap (2)

Objective Function

$$\mathcal{L}(w_0, w_1) = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - (w_0 + w_1 x_i)\right)^2$$



$$f(x_i) = w_0 + w_1 x_i$$

$$e_i = y_i - f(x_i)$$

$$\mathcal{L}(w_0, w_1) = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - (w_0 + w_1 x_i)\right)^2$$

$$\underset{w_0, w_1}{\operatorname{argmin}} \; \mathcal{L}(w_0, w_1)$$

Thus, it is convex, at the unique minimum of our loss function, its "partial" derivative with respect to w_0 and w_1 will be zero!

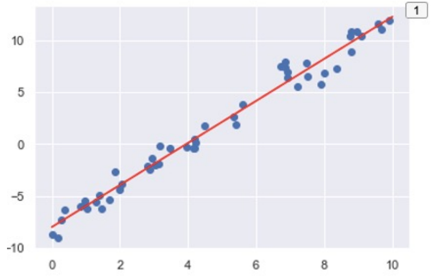
The Least Square Solution

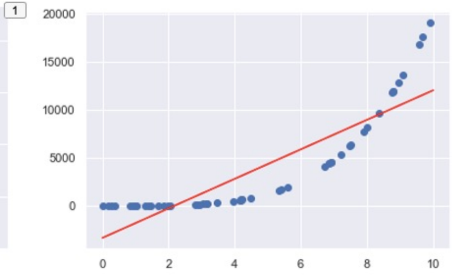
$$\mathcal{L}(w_0, w_1) = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - (w_0 + w_1 x_i)\right)^2$$

1. Compute partial derivatives of the loss function with respect to w_0 and w_1
2. Set them to 0
3. And solve for w_0 and w_1

$$\boxed{\begin{aligned} w_0 &= \overline{y} - w_1\overline{x} \\ w_1 &= \frac{\overline{xy} - \overline{x}\,\overline{y}}{\overline{x^2} - (\overline{x})^2} \end{aligned}}$$

Recap (3)





Polynomial Regression

- Using the same framework that we learned, to fit a family of more complex models through a <u>transformation of predictors</u>

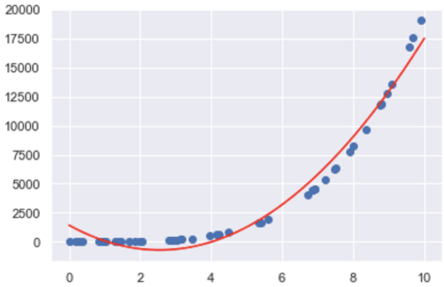
- That is,

$$y = w_0 + w_1 x + w_2 x^2$$

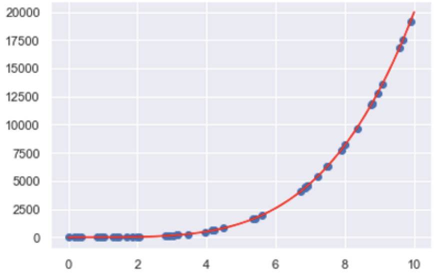
- More generally,

$$y = w_0 + w_1 x + w_2 x^2 + \cdots + w_d x^d$$

- Do not forget, "the model is still linear in parameters"



Order or Degree 2



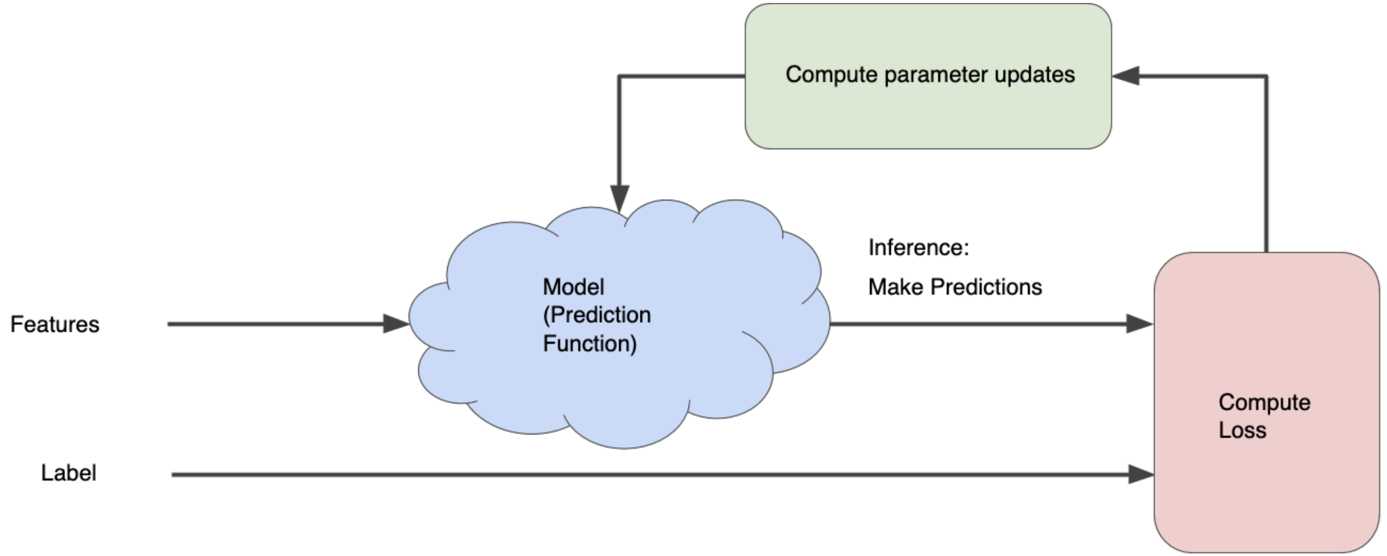
Order or Degree 4

Gradient Descent

- Learning the machine learning model by iteratively reducing the loss

- More like how we learn: *learn by making mistakes*

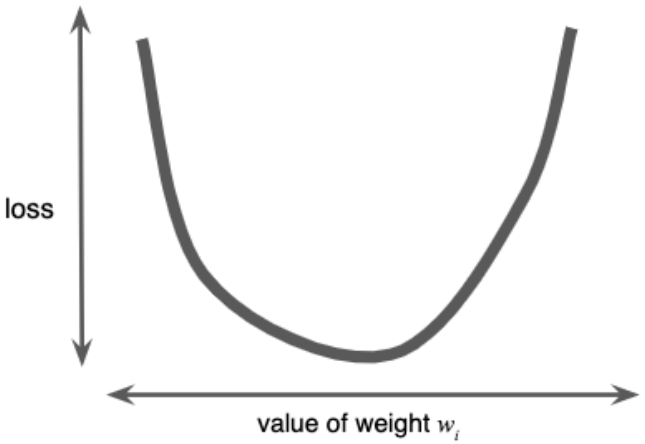
Iteratively Reducing the Loss



https://developers.google.com/machine-learning/crash-course/reducing-loss/an-iterative-approach

Mean Squared Error

$$\mathcal{L}(w_0, w_1) = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - (w_0 + w_1 x_i)\right)^2$$

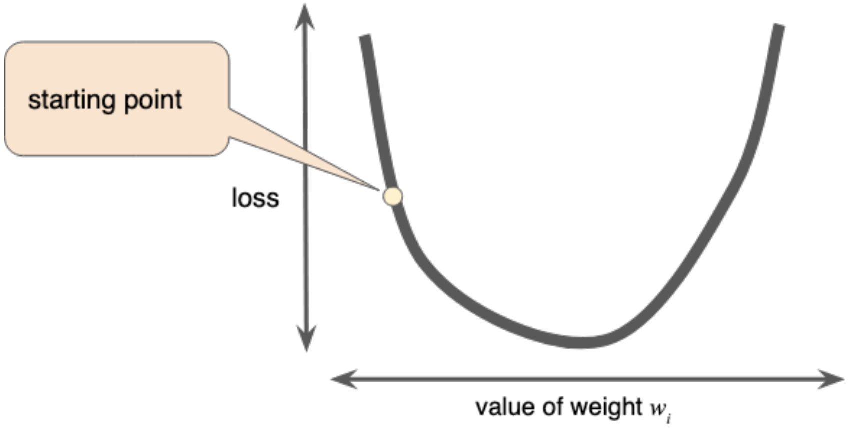


Regression problems yield convex loss vs. weight plots

https://developers.google.com/machine-learning/crash-course/reducing-loss/gradient-descent

Gradient Descent (1)

- Pick a starting value of w_i



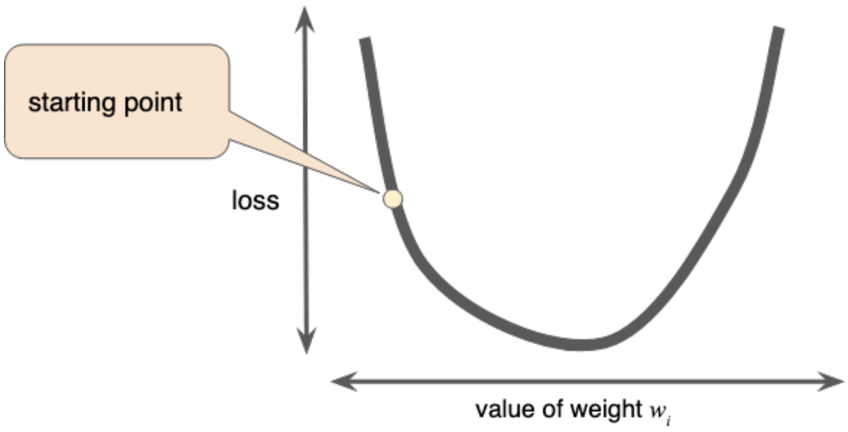
A starting point for gradient descent.

https://developers.google.com/machine-learning/crash-course/reducing-loss/gradient-descent

Gradient Descent (2)

- Compute the *gradient* of the loss curve at the starting point

- Gradient of the curve at any point is equal to the *derivative* of the curve at that point



A starting point for gradient descent.

https://developers.google.com/machine-learning/crash-course/reducing-loss/gradient-descent

Gradient Descent (3)

- Gradient is a vector: has both magnitude and direction

- It always points to the direction of the steepest increase in the loss function

- What is our objective?
  - ➢ Reduce the loss

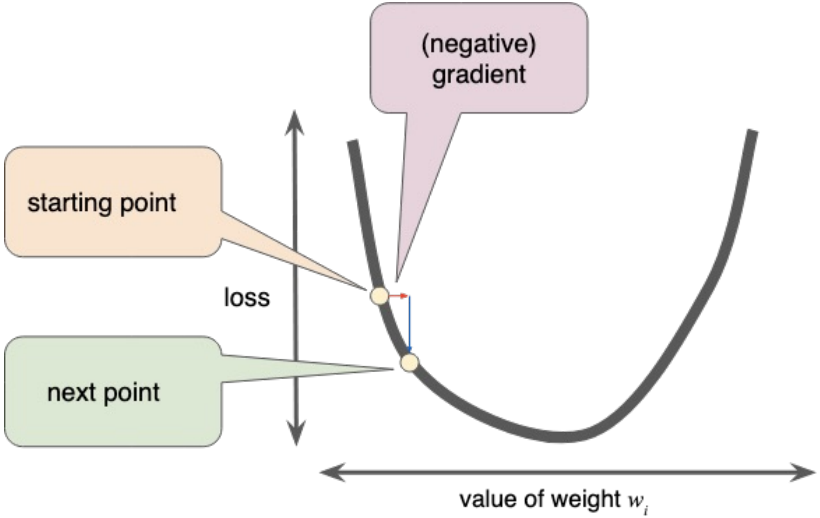
➢ Thus we can reduce the loss by taking a step in the direction of negative gradient

Gradient Descent (4)

- Reduce the loss by taking a step in the direction of negative gradient

- How much should we move in the desired direction?

    - ➢ some fraction of the gradient's magnitude

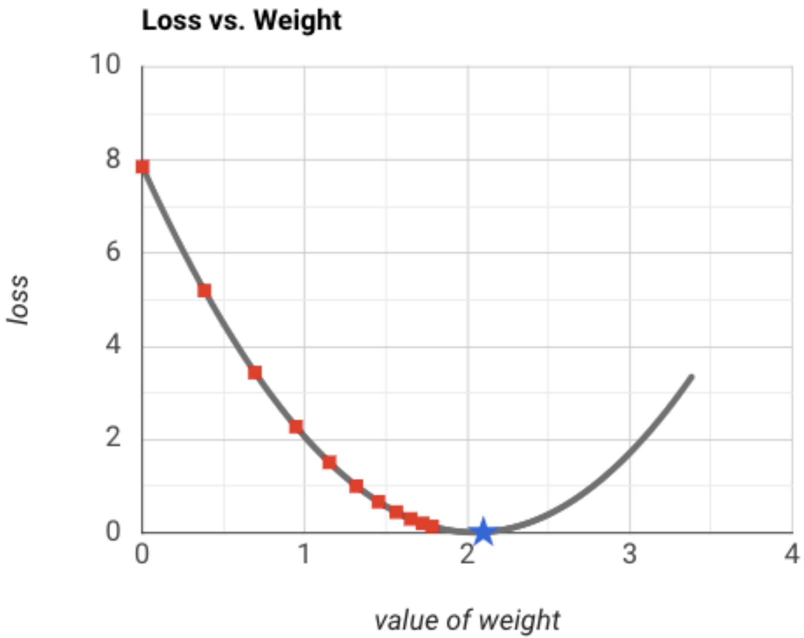


A gradient step moves us to the next point on the loss curve

https://developers.google.com/machine-learning/crash-course/reducing-loss/gradient-descent

Gradient Descent (5)

- The gradient descent then repeats this process, edging ever closer to the minimum.



https://developers.google.com/machine-learning/crash-course/fitter/graph

More Formally

$$\mathcal{L}(w_0, w_1) = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - (w_0 + w_1 x_i)\right)^2 = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

- Gradient descent reduces the loss function iteratively

- This is done by updating the values of the parameters as

$$w_0 = w_0 - \alpha \frac{\partial \mathcal{L}}{\partial w_0}$$

$$w_1 = w_1 - \alpha \frac{\partial \mathcal{L}}{\partial w_1}$$

➢some fraction of the gradient's magnitude

Known as the *learning rate*

Partial Derivatives of $\mathcal{L}(w_0, w_1)$ (1)

$$\mathcal{L}(w_0, w_1) = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - (w_0 + w_1 x_i)\right)^2$$

- Let's make a slight modification to the loss function just to make the math easier, later on

$$\mathcal{L}(w_0, w_1) = \frac{1}{2n}\sum_{i=1}^{n}\left(y_i - (w_0 + w_1 x_i)\right)^2$$

Partial Derivatives of $\mathcal{L}(w_0, w_1)$ (2)

$$\mathcal{L}(w_0, w_1) = \frac{1}{2n}\sum_{i=1}^{n}\left(y_i - (w_0 + w_1 x_i)\right)^2$$

- Let's expand the left hand side

$$\mathcal{L}(w_0, w_1) = \frac{1}{2n}\sum_{i=1}^{n}\left(w_1^2 x_i^2 + 2w_1 x_i w_o - 2w_1 x_i y_i + w_0^2 - 2w_0 y_i + y_i^2\right)$$

$$\frac{\partial \mathcal{L}}{\partial w_0}(1)$$

$$\mathcal{L}(w_0, w_1) = \frac{1}{2n}\sum_{i=1}^{n}\left(w_1^2 x_i^2 + 2w_1 x_i w_o - 2w_1 x_i y_i + w_0^2 - 2w_0 y_i + y_i^2\right)$$

- Let's simplify by removing the terms that do not involve w_0

$$= \frac{1}{2n}\sum_{i=1}^{n}(2w_1 x_i w_o + w_0^2 - 2w_0 y_i)$$

- Now, we are ready to compute the partial derivative

$$\frac{\partial \mathcal{L}}{\partial w_0} = \frac{1}{2n}\sum_{i=1}^{n}(2w_1 x_i + 2w_0 - 2y_i)$$

$$\frac{\partial \mathcal{L}}{\partial w_0}(2)$$

$$\frac{\partial \mathcal{L}}{\partial w_0} = \frac{1}{2n}\sum_{i=1}^{n}(2w_1x_i + 2w_0 - 2y_i)$$

- The 2's cancel out (see, that's why we included it earlier)

$$\frac{\partial \mathcal{L}}{\partial w_0} = \frac{1}{n}\sum_{i=1}^{n}(w_1x_i + w_0 - y_i)$$

- We can rewrite it as

$$\frac{\partial \mathcal{L}}{\partial w_0} = \frac{1}{n}\sum_{i=1}^{n}(\widehat{y}_i - y_i)$$

$$\frac{\partial \mathcal{L}}{\partial w_1}$$

$$\mathcal{L} = \frac{1}{2n}\sum_{i=1}^{n}\left(w_1^2 x_i^2 + 2w_1 x_i w_o - 2w_1 x_i y_i + w_0^2 - 2w_0 y_i + y_i^2\right)$$

- Next we will calculate the partial derivative of the loss with respect to w_1
- You will do it yourself: apply the same procedure we used for w_0
- The outcome should be

$$\frac{\partial \mathcal{L}}{\partial w_1} = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)\, x_i$$

Thus

$$\mathcal{L}(w_0, w_1) = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - (w_0 + w_1 x_i)\right)^2 = \frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{y}_i)^2$$

- Gradient descent reduces the loss function iteratively.

- This is done by updating the values of the parameters as

$$w_0 = w_0 - \alpha\frac{\partial\mathcal{L}}{\partial w_0} = w_0 - \alpha\frac{1}{n}\sum_{i=1}^{n}(\widehat{y}_i - y_i)$$

$$w_1 = w_1 - \alpha\frac{\partial\mathcal{L}}{\partial w_1} = w_1 - \alpha\frac{1}{n}\sum_{i=1}^{n}(\widehat{y}_i - y_i)\, x_i$$

GD Summary (Single Predictor)

Repeat {

$$w_0 = w_0 - \alpha \frac{1}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i)$$

$$w_1 = w_1 - \alpha \frac{1}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i)\, x_i$$

}

Multiple Predictors

- In real-life, we most deal with situations where a response variable depends on multiple predictors

$$y = w_0 + w_1 x_1 + w_2 x_2 + \cdots w_9 x_9$$

- Luckily, this requires a simple update to GD

Recall: GD Summary (Single Predictor)

Repeat {

$$w_0 = w_0 - \alpha \frac{1}{n} \sum_{i=1}^{n} (\widehat{y_i} - y_i)$$

$$w_1 = w_1 - \alpha \frac{1}{n} \sum_{i=1}^{n} (\widehat{y_i} - y_i)\, x_i$$

}

I can Rewrite it as

Repeat {

$$w_0 = w_0 - \alpha \frac{1}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i)\, x_i^0$$

$$w_1 = w_1 - \alpha \frac{1}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i)\, x_i^1$$

}

GD Summary (Multiple Predictors)

Repeat {

$$w_j = w_j - \alpha \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)\, x_i^j$$

}

Classification

- Response is qualitative, discrete or categorical.

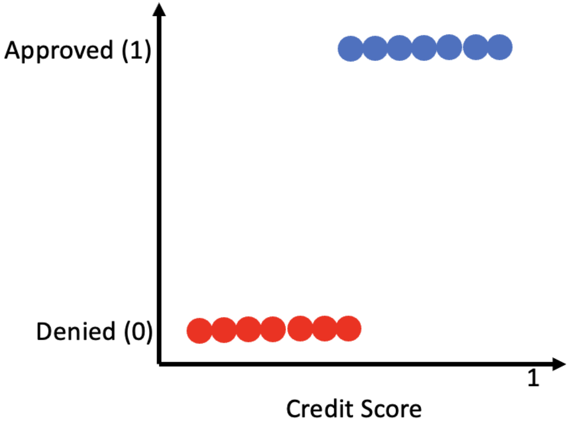
- For examples,
  - Eye color ∈ $\{brown, blue, green\}$
  - Email ∈ $\{Spam, Ham\}$
  - Expression ∈ $\{Happy, Sad, Angry\}$
  - Action ∈ $\{Walking, Running, Cycling\}$

Binary Classification

- Classification task that has two class labels

- For example,
  - People whose credit applications were either approved or denied

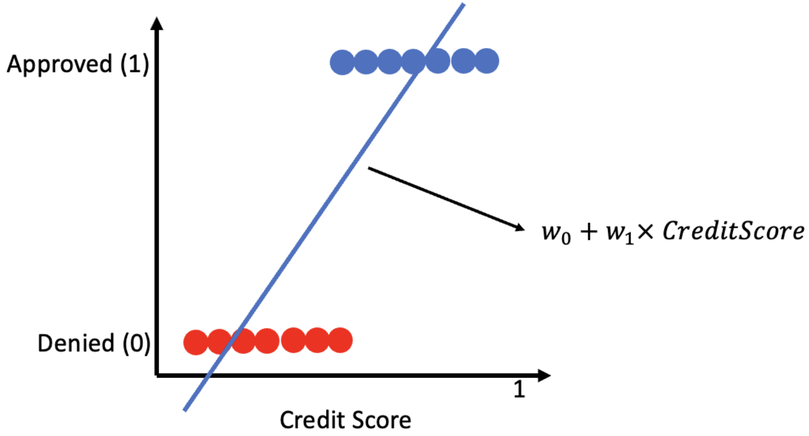
  - For simplicity, we are using a single feature

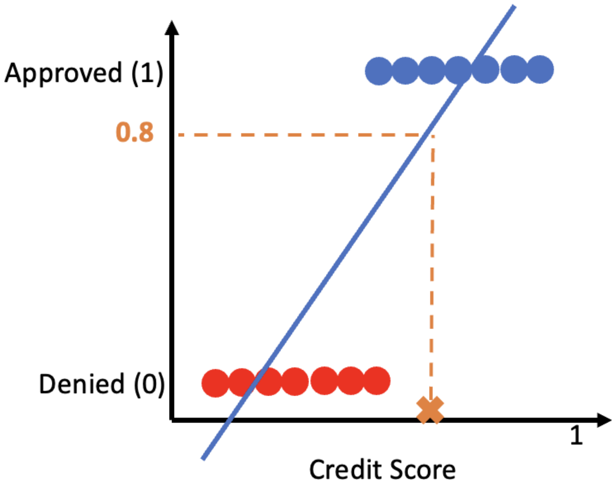


What if we Treat the Problem As Predicting Class Probability?

- Instead of predicting the class of a data point, what if we computed the probability of the data point belonging to a certain class?
- That is $p(Approved = Yes \mid creditscore)$
- We can “threshold” the result to decide the class
- What can be the advantage of doing this?
  - y becomes continuous
  - Thus, we can solve this as a regression problem (reuse of what we learned earlier)

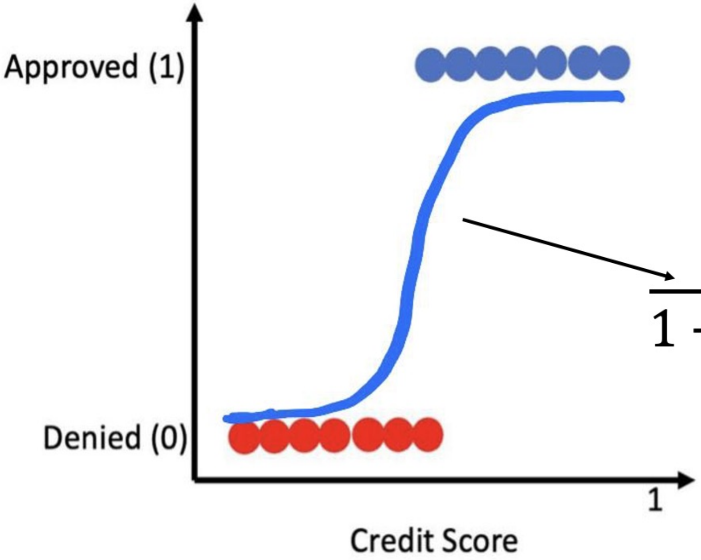
Using Regression for Probability Estimation





Do you see any problems?

Restricting the Output to be b/w 0 and 1

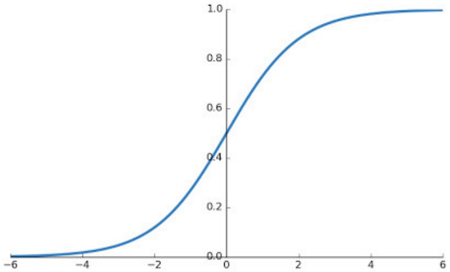


$$\frac{1}{1 + e^{-(w_0 + w_1 \times CreditScore)}}$$

Logistic Regression

- Thus, we must model $p(x)$ using a function that gives outputs between 0 and 1 for all values of x.

- One such function is the *Logistic or sigmoid function*



$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

- Thus, in logistic regression, we use the **Logistic Function**

Logistic Regression (Single Predictor)

$$p(x_1) = \frac{1}{1 + e^{-z}}$$

$$z = w_0 + w_1 x_1$$

Logistic Regression (Multiple Predictors)

$$p(\boldsymbol{x}) = \frac{1}{1 + e^{-z}}$$

$$z = w_0 + w_1 x_1 + \cdots + w_p x_p$$

Estimating Parameters of Logistic Regression

$$p(\boldsymbol{x}) = \frac{1}{1 + e^{-z}}$$

$$z = w_0 + w_1 x_1 + \cdots + w_p x_p$$

We need a *loss function*

A Short Detour

- Linear sum as a dot product

$$y = w_0 + w_1 x_1 + w_2 x_2 + \cdots w_d x_d$$

$$\boldsymbol{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \end{bmatrix} \qquad \boldsymbol{x} = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_d \end{bmatrix}$$

$$y = \boldsymbol{w}^T . \boldsymbol{x}$$

Loss Function of Logistic Regression

$$y = \frac{1}{1 + e^{-\boldsymbol{w}^T \boldsymbol{x}^i}}$$

- First take a look at linear regression loss function

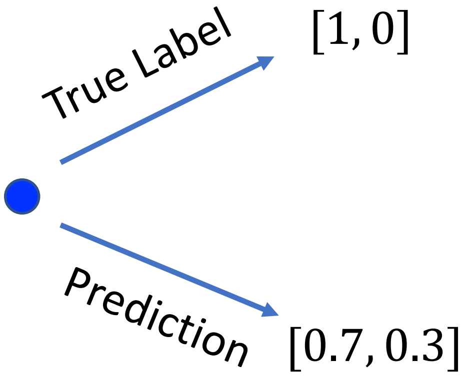
$$\mathcal{L}(\boldsymbol{w}) = \frac{1}{n} \sum_{i=1}^{n} \left(y^i - \boldsymbol{w}^T \boldsymbol{x}^i \right)^2$$

- Due to non-linearity of the sigmoid function in hypothesis of Logistic Regression, MSE is **not convex** anymore

Thus we need a new Loss Function

Loss Function for Logistic Regression-1

- Imagine a positive example in our credit approval application



Thus we have two **probability distributions** for each training example: one is the true distribution and the other is the distribution that the model predicts for the training example

Loss Function for Logistic Regression-2

- Thus we have two probability distributions

- In order to see how well our model is doing, we can measure the difference between these two distributions, and this is where cross entropy comes into play.

Loss Function for Logistic Regression-3

Cross Entropy

- is a measure of the difference between two probability distributions. The cross-entropy loss between two probability distributions t and p is defined as:

  H(t, p) = - ∑ t(i) * log(p(i)), where i is indexed over classes

- For t = [1, 0] and p = [0.7, 0.3]

  H(t, p) = - (1 * log(0.7) + 0 * log(0.3))
  = - (- 0.35667494 + 0)
  = 0.35667494

- For t = [1, 0] and p = [0.9, 0.1]

  H(t, p) = - (1 * log(0.9) + 0 * log(0.1))
  = - (- 0.105360515 + 0)
  = 0.105360515

Loss Function for Logistic Regression-4

Cross Entropy in a binary classification problem

The formula for binary cross-entropy between the predicted probability, p, and the true label, y, is given by:

Binary Cross-Entropy Loss = - (y * log(p) + (1 - y) * log(1 - p))

Where:

- y is the true label (0 or 1)
- p is the predicted probability of the positive class (usually between 0 and 1)
- log is the natural logarithm function

Loss Function of Logistic Regression (5)

$$\mathcal{L}(\boldsymbol{w}) = -\frac{1}{n}\sum_{i=1}^{n} y^i \log\left(p(x^i)\right) + (1 - y^i)\log(1 - p(x^i))$$

- Given this loss function,
- We will define our objective function as

$$\boxed{\underset{w_o, w_1}{\operatorname{argmin}}\ \mathcal{L}(\boldsymbol{w})}$$

Solving Our Objective

- Given our objective function, we solve it using

  - **Gradient Descent**

- Thus, what we need to do is to derive the equation for parameter update

$$w_j = w_j - \alpha \frac{\partial \mathcal{L}}{\partial w_j}$$

A Short Detour, again!

- We will use the following facts, note $\sigma(.)$ is the logistic or sigmoid function

$$\frac{d\big(\sigma(x)\big)}{dx} = \sigma(x)\big(1 - \sigma(x)\big)$$

$$\frac{d\big(log(x)\big)}{dx} = \frac{1}{x}\frac{d}{dx}x = \frac{1}{x}$$

$$\frac{d\Big(f\big(g(x)\big)\Big)}{dx} = f'\big(g(x)\big)g'(x)$$

Deriving the Equation for Parameter Update

$$\mathcal{L} = -\frac{1}{n}\sum_{i=1}^{n} y_i \log\big(p(x_i)\big) + (1 - y_i)\log(1 - p(x_i))$$

$$\mathcal{L} = -\frac{1}{n}\sum_{i=1}^{n} y_i \log\big(\sigma(z_i)\big) + (1 - y_i)\log(1 - \sigma(z_i))$$

$$z_i = \boldsymbol{w}^T \boldsymbol{x}^i = w_0 + w_1 x_1^i + \cdots + w_p x_p^i$$

Deriving the Equation for Parameter Update (2)

$$\mathcal{L} = -\frac{1}{n}\sum_{i=1}^{n} y_i \boxed{\log\big(\sigma(z_i)\big)} + (1-y_i)\boxed{\log(1-\sigma(z_i))}$$

- Computing $\frac{\partial \mathcal{L}}{\partial w_j}$
- These are what depend on w_j
- So let's compute their partial derivative w.r.t w_j

Deriving the Equation for Parameter Update (3)

$$\frac{\partial \log(\sigma(z_i))}{\partial w_j} = \frac{1}{\sigma(z_i)} \frac{\partial \sigma(z_i)}{\partial z_i} \frac{\partial z_i}{\partial w_j}$$

- Which rules did we apply?
  - Rule of log
  - Chain rule

Deriving the Equation for Parameter Update (4)

$$\frac{\partial \log(\sigma(z_i))}{\partial w_j} = \frac{1}{\sigma(z_i)} \frac{\partial \sigma(z_i)}{\partial z_i} \frac{\partial z_i}{\partial w_j}$$

$$= \frac{1}{\cancel{\sigma(z_i)}} \cancel{\sigma(z_i)}(1 - \sigma(z_i)) \frac{\partial z_i}{\partial w_j}$$

- Which rule did we apply?
  - Rule of sigmoid function

Deriving the Equation for Parameter Update (5)

$$\frac{\partial \log\left(\sigma(z_i)\right)}{\partial w_j} = \left(1 - \sigma(z_i)\right)\frac{\partial z_i}{\partial w_j}$$

$$z_i = \boldsymbol{w}^T \boldsymbol{x}^i = w_0 + w_1 x_1^i + \cdots + w_p x_p^i$$

$$\frac{\partial z_i}{\partial w_j} = x_j^i$$

$$\boxed{\frac{\partial \log\left(\sigma(z_i)\right)}{\partial w_j} = \left(1 - \sigma(z_i)\right) x_j^i}$$

Recall

$$\mathcal{L} = -\frac{1}{n}\sum_{i=1}^{n} y_i \boxed{\log\big(\sigma(z_i)\big)} + (1-y_i)\boxed{\log(1-\sigma(z_i))}$$

- Computing $\frac{\partial \mathcal{L}}{\partial w_j}$
- These are what depend on w_j
- So let's compute their partial derivative w.r.t w_j

Deriving the Equation for Parameter Update (6)

$$\frac{\partial \log(1 - \sigma(z^i))}{\partial w_j} = ?$$

- You will do this activity yourselves
- You just have to apply the same rules

Deriving the Equation for Parameter Update (7)

$$\frac{\partial \log(1-\sigma(z^i))}{\partial w_j} = -\sigma(z^i)x_j^i$$

Deriving the Equation for Parameter Update (8)

$$\mathcal{L} = -\frac{1}{n}\sum_{i=1}^{n} y_i \boxed{\log\big(\sigma(z_i)\big)} + (1 - y_i)\boxed{\log(1 - \sigma(z_i))}$$

$$\frac{\partial \log\big(\sigma(z_i)\big)}{\partial w_j} = \big(1 - \sigma(z_i)\big)x_j^i$$

$$\frac{\partial \log\big(1 - \sigma\big(z^i\big)\big)}{\partial w_j} = -\sigma\big(z^i\big)x_j^i$$

Deriving the Equation for Parameter Update (8)

$$\mathcal{L} = -\frac{1}{n}\sum_{i=1}^{n} y_i \boxed{\log(\sigma(z_i))} + (1 - y_i)\boxed{\log(1 - \sigma(z_i))}$$

- Thus

$$\frac{\partial \mathcal{L}}{\partial w_j} = -\frac{1}{n}\sum_{i=1}^{n} y^i \left(1 - \sigma(z^i)\right) x_j^i - (1 - y^i)\sigma(z^i)x_j^i$$

Deriving the Equation for Parameter Update (9)

$$\frac{\partial \mathcal{L}}{\partial w_j} = -\frac{1}{n}\sum_{i=1}^{n} y^i \left(1 - \sigma\left(z^i\right)\right) x_j^i - \left(1 - y^i\right)\sigma\left(z^i\right)x_j^i$$

- Solving it further by multiplying and simplifying, we will get

$$\frac{\partial \mathcal{L}}{\partial w_j} = -\frac{1}{n}\sum_{i=1}^{n} \left(y^i - \sigma\left(z^i\right)\right) x_j^i$$

Deriving the Equation for Parameter Update (10)

$$\frac{\partial \mathcal{L}}{\partial w_j} = -\frac{1}{n}\sum_{i=1}^{n}\left(y^i - \sigma\left(z^i\right)\right) x_j^i$$

- We can rewrite it as

$$\frac{\partial \mathcal{L}}{\partial w_j} = \frac{1}{n}\sum_{i=1}^{n}\left(p\left(x^i\right) - y^i\right) x_j^i$$

Deriving the Equation for Parameter Update (11)

- Thus,

$$w_j = w_j - \alpha \frac{1}{n} \sum_{i=1}^{n} \left(p(x^i) - y^i \right) x_j^i$$

Takeaway

In logistic regression the training process works as follows:

1. Initialise the model parameters with random values.
2. Feed the training data through the model, and use the model's predictions to calculate the cross-entropy loss between the predicted probabilities and the true labels.
3. Calculate the gradient of the loss function with respect to the model parameters.
4. Update the model parameters in the opposite direction of the gradient, using a small learning rate. This will 'step' the model in the direction that decreases the loss.
5. Repeat steps 2-4 until the model converges, or until a maximum number of iterations is reached.
6. During training, the model will 'learn' the optimal parameters that minimise the cross-entropy loss on the training data. Once training is complete, the model can be used to make predictions on new, unseen examples.

Note

- We move over all data, average the result and then update the parameter

$$w_j = w_j - \alpha \boxed{\frac{1}{n}\sum_{i=1}^{n}} \left(p(x^i) - y^i \right) x_j^i$$

Types of Gradient Descent

- Batch GD
- Stochastic GD
- Mini-batch GD

From Probability to Class Label

$$\hat{y} = \begin{cases} 1, & \hat{p}(\boldsymbol{x}) > Th \\ 0, & otherwise \end{cases}$$

Metrics (1)

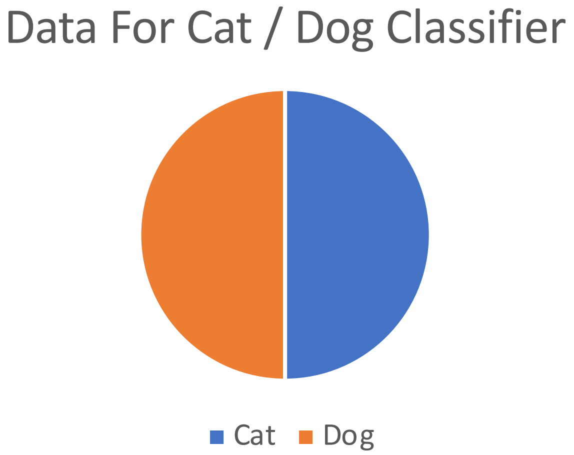
- Accuracy

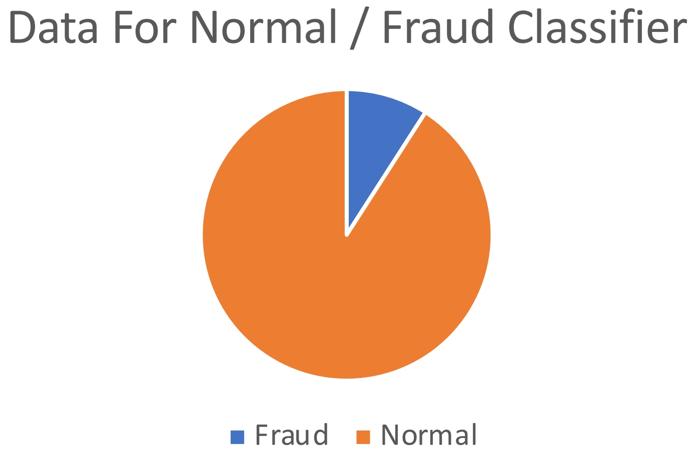
$$Acc = 1 - \frac{\#\ of\ missclassified\ examples}{\#\ of\ samples}$$

- But sometimes, accuracy is not a good metric to use to measure performance of machine learning models

Things to Know about Machine Learning

- Class-imbalance





Example

Confusion Matrix

1. How accurate is this classifier?

Predicted Class Label

Actual Class Label		Negative	Positive
	Negative	998	0
	Positive	2	0

2. But what if the missclassified data points (False Negatives) are someone with a serious disease, or a serious fraudalent transcation, or a bad malware, etc.

General Requirements for ML Models

1. Better than random guessing

2. Better than majority guessing

Confusion Matrix

		Predicted Class Label	
		Negative	**Positive**
Actual Class Label	Negative	True Negatives	False Positives
	Positive	False Negatives	True Positives

Metrics (2)

		Predicted Class Label	
		Negative	Positive
Actual Class Label	Negative	True Negatives	False Positives
	Positive	False Negatives	True Positives

1. Precision

$$\textit{Precision} = \frac{\textit{True Positives}}{\textit{True Positives} + \textit{False Positives}}$$

2. Recall

$$\textit{Recall} = \frac{\textit{True Positives}}{\textit{True Positives} + \textit{False Negatives}}$$

Back to Our Example

		Predicted Class Label	
		Negative	Positive
Actual Class Label	Negative	998	0
	Positive	2	0

1. Precision = 0

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

2. Recall = 0

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

Our Example–Case 2

		Predicted Class Label	
		Negative	Positive
Actual Class Label	Negative	998	0
	Positive	0	2

1. Precision = 1

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

2. Recall = 1

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

Our Example–Case 3

		Predicted Class Label	
		Negative	Positive
Actual Class Label	Negative	998	0
	Positive	1	1

1. Precision = 1

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

2. Recall = 0.5

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

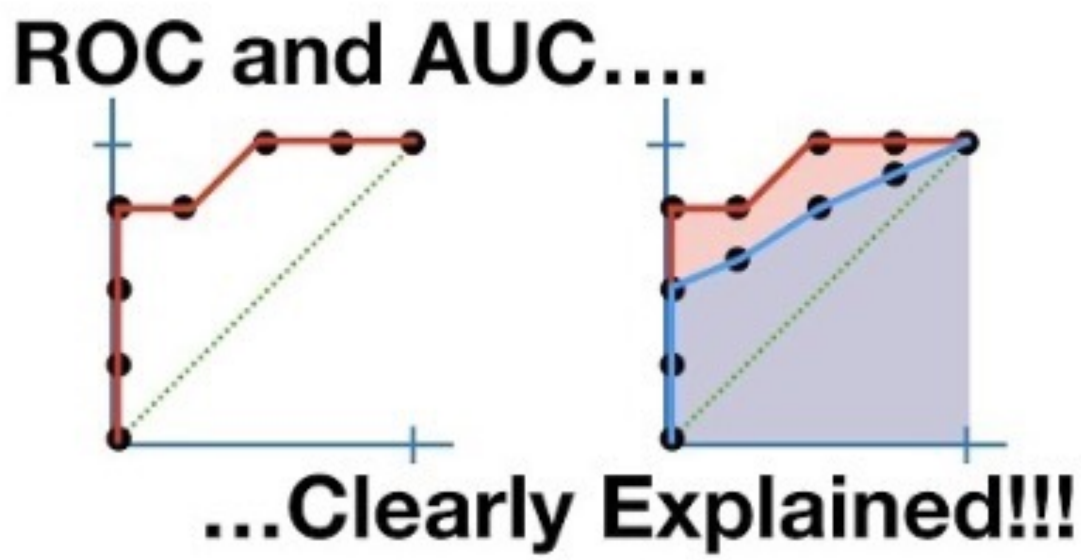
Balancing Precision and Recall

- F1 Score

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Imp: Self-learning

- Watch the following video Receiver Operator Characteristic (ROC) graphs and the area under the curve (AUC), and see how it can be used for thresholding



Summary

- Gradient Descent
- Classification Tasks
- Logistic Regression and its learning objective
- Solving the objective
  - Driving the equation for parameter updates
  - Gradient Descent
- Classification metrics and class imbalance

Reflection

- Why is predicting whether a software is malware or not-a-malware a classification problem?
- What do the words “logistic”, and “regression” represent in Logistic Regression?
- Why do we want to minimise the cross entropy loss with respect to the parameters of our logistic regression model?
- What is thresholding and why is it important?
- What is ROC curve and how can we use it for thresholding?