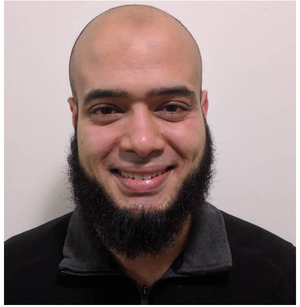
Machine Learning

Binary Classification Cost Function

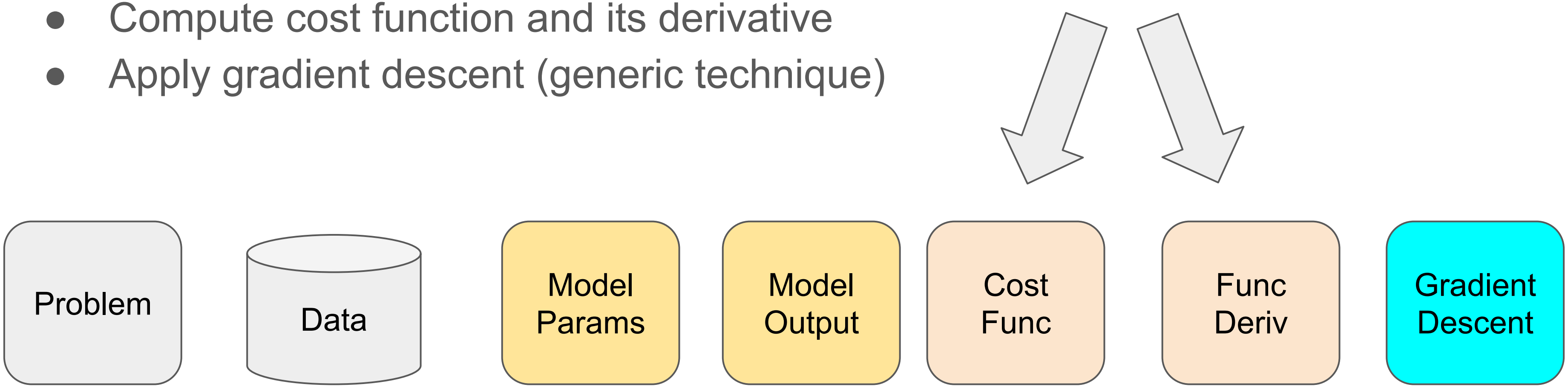
Mostafa S. Ibrahim
Teaching, Training and Coaching for more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / MSc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)



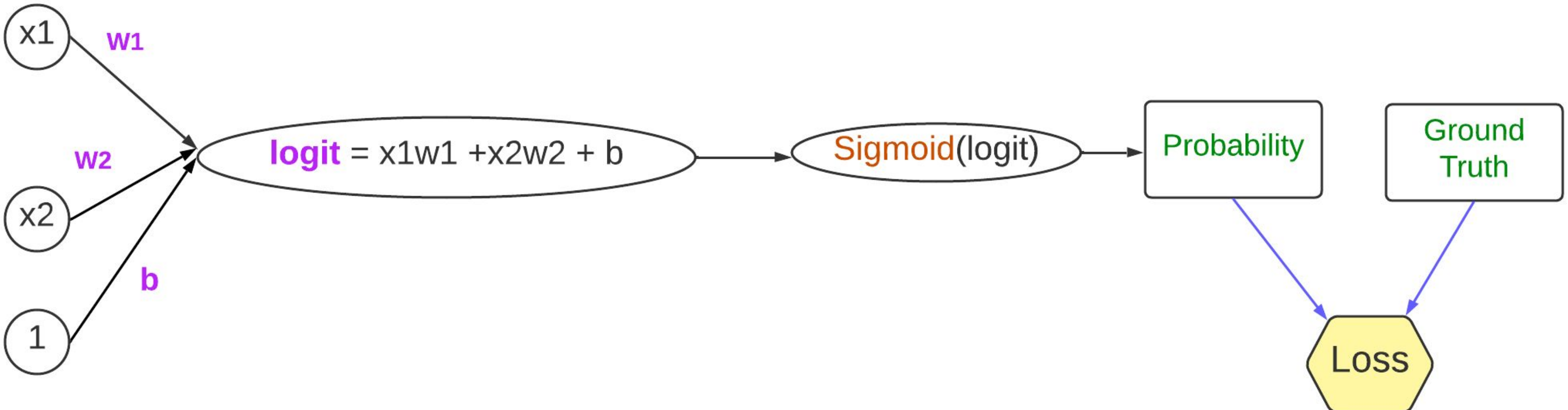
Recall Modeling Style So Far

- Define a problem and collect its data
- Select a suitable model
- Given data and the model, we get the model output
- Compute cost function and its derivative
- Apply gradient descent (generic technique)



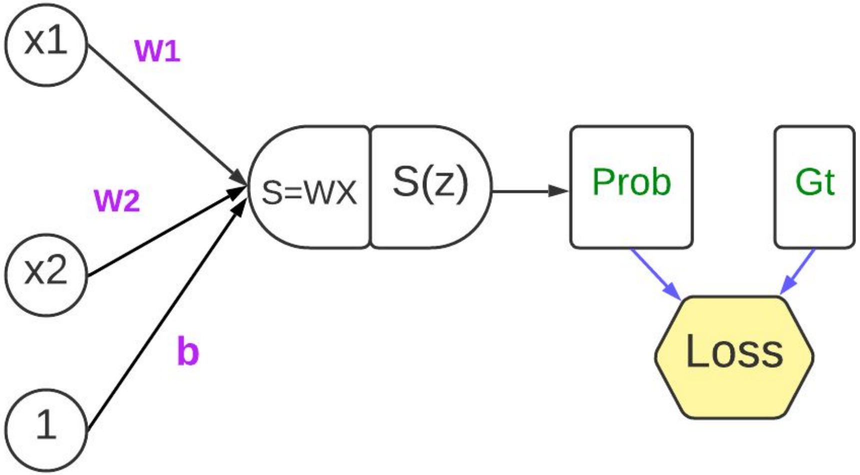
So far

- We just extended the linear regression with a sigmoid step
  - We showed this maps output to [0-1], which can be interpreted as probability
  - We understand the logit equation represents a decision boundary
- However, to train the model, e.g. using gradient descent, we need **a cost function** between the output (probability) and ground truth!



Simplifying the drawing

- Let's use Neural Network notation
- We aggregate WX+B, the logit and then apply sigmoid function S(x) to generate a probability
- Loss function is computed between the probability and the ground truth (gt)



Recall MSE cost for Linear Regression

- LR formulation represents a **convex** function

$$y(X^n, W) = \sum_{j=0}^{d} X_j^n W_j$$

$$cost(W) = \frac{1}{2N} \sum_{n=1}^{N} (y(X^n, W) - t^n)^2$$

$$\frac{\partial cost(W)}{\partial W_j} = \frac{1}{N} \sum_{n=1}^{N} (y(X^n, W) - t^n) * X_j^n$$

MSE cost for Logistic Regression

- In logistic, y(X, W) = sigmoid(X^TW)
- The major trouble now is this function is **not convex** anymore
  - Intuitively, this complex non-linear mapping may make it non-convex [proof claim]
  - Most of what we face is non-convex
- Another concern: it is hard to extend multi-class classification (coming)

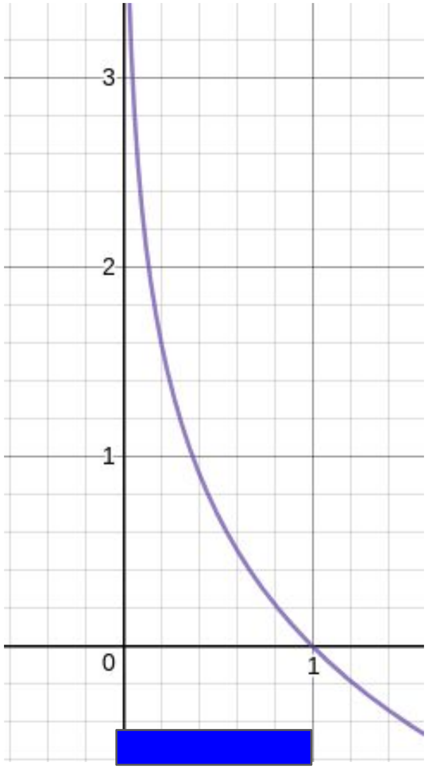
$$cost(W) = \frac{1}{2N}\sum_{n=1}^{N}(y(X^n, W) - t^n)^2$$

Log Loss

- The most common cost function for classification. Also known as:
  - Binary Cross-Entropy Loss / **negative** log likelihood loss / Logistic loss
  - It can be derived based on maximum likelihood estimation
  - We will visit again for the general multiclassifier case
- Assume $p^{(n)}$ is the predicted probability of the input to be 1: p(1 | x)
- Assume $\mathbf{t^{(n)}}$ **is the ground truth** for an example {either 0 or 1}
- We have 2 cases
  - If t = 1, then use the cost function -log(p)
  - If t = 0, then use the cost function -log(1 - p)
    - 1 - pi gives us the probability of output 0: p(0 | x)
- In other words: it is a loss function with 2 branches based on $t^{(n)}$
  - We sum the error of the positive inputs and sum for the negative inputs

Log Loss: t = 1

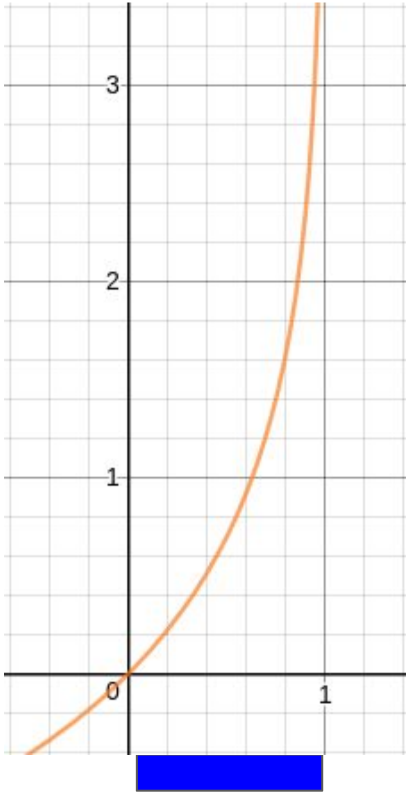
- If t = 1, then use the cost function -log(p)
- Recall: p is probability in range [0, 1]
- Then reduce -log(p) for only [0, 1]
- Intuitively:
  - we want if p ~= 1, we get penalty 0
  - we want if p ~= 0, we get very high penalty
- Look to the curve
  - Starting from p=1, the error is zero and then increasing



-ln(p)

Log Loss: t = 0

- If t = 0, then use the cost function -log(1 - p)
- Again, we care only about [0, 1] range
- Opposite logic - Intuitively:
  - we want if p ~= 0, we get penalty 0
  - we want if p ~= 1, we get very high penalty
- Look to the curve
  - Starting from p=0, the error is zero and then increasing

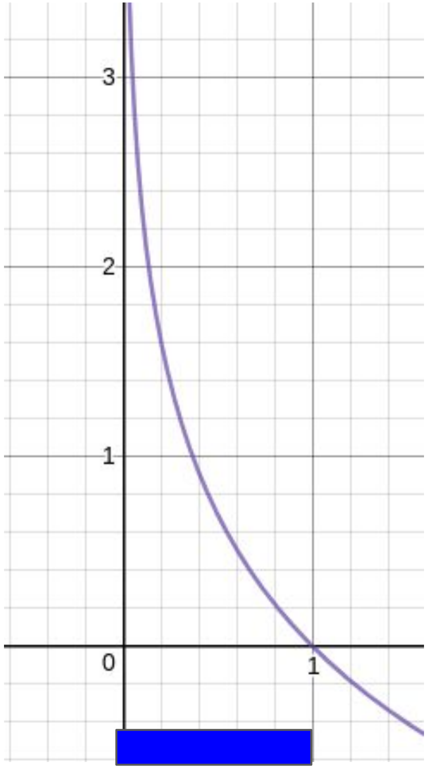


-ln(1 - p)

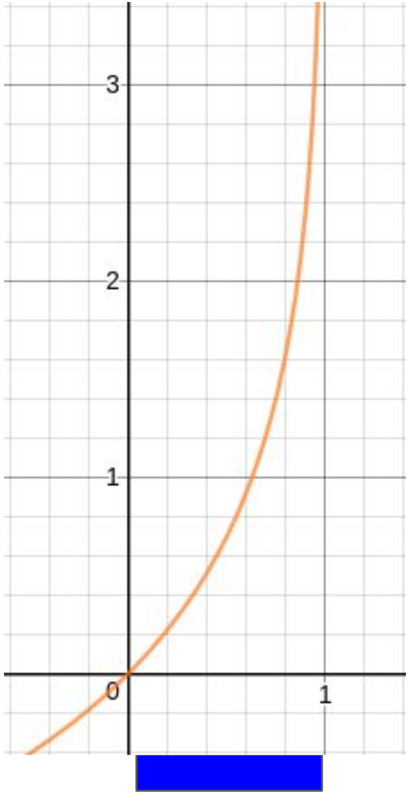
Merging the 2 cases

- We can write a single function that just merges the 2 cases
    - Note if t = 1, then 1-t = 0 [activate the first part]
    - Note if t = 0, then 1-t = 1 [activate the second part]
    - That is; one branch only is active
- We can prove this function is convex
    - Let's show visual intuition

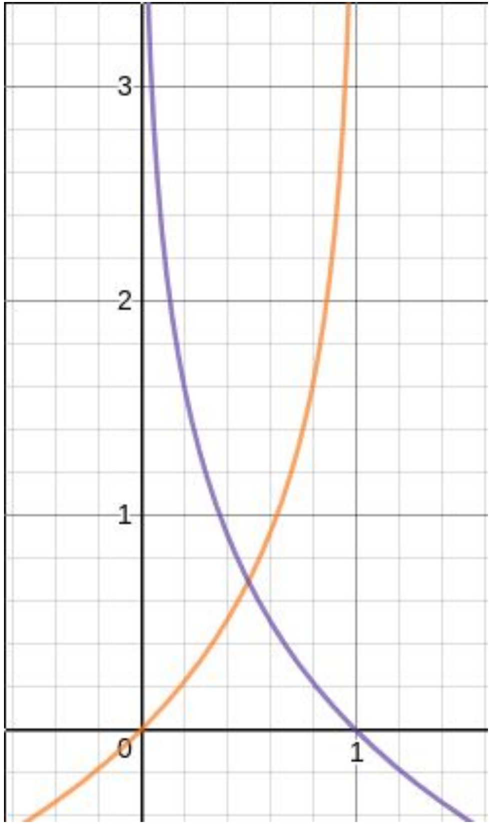
$$logloss(p,t) = -tlog(p) - (1-t)log(1-p)$$



-ln(x)



-ln(1-x)



-ln(x) - ln(1-x)

Convex]0, 1[

Big Picture so far

- We wanted to solve the classification problem
- We transformed the output to range [0-1] using sigmoid
- We defined the cost function using logloss
- Now we just minimize then using gradient descent!
    - New_x = old_x - gradient x learning_rate
- To do, we need to compute the partial derivatives of the cost function

Big Picture so far

$$y(X^n, W) = p^n = Sigmoid(W^T * X^n)$$

$$cost(W) = \frac{1}{N} \sum_{n=1}^{N} (logloss(y(X^n, W), t^n))$$

$$logloss(p, t) = -tlog(p) - (1 - t)log(1 - p)$$

Logloss Derivative

- Now we compute the derivative of a one example (then average examples)
- It is direct chain rule application
- Logloss is function of p which is sigmoid of z which is linear equation W^TX
- ∂Logloss/∂w1 = ∂Logloss/∂p * p/∂z * ∂x/∂w1
  - Compute this equation for each branch and merge
- ∂Logloss/∂w1 = **(p - t) * x1**
  - This is identical to linear regression, just p = sigmoid(linear equation)!

$$logloss(p,t) = -t\,log(p) - (1-t)\,log(1-p)$$

Compute Derivatives - Reading Homework

$$\frac{d}{dz}\sigma(z) = \sigma(z) \cdot (1 - \sigma(z))$$

$$p = \sigma(z) = \frac{1}{1 + e^{-z}} = \frac{1}{1 + e^{-(\mathbf{w}^\top \mathbf{x} + b)}}$$

$$\frac{d}{dp}\log(p) =$$

$$\frac{d}{dp}\log(1 - p) =$$

$$\frac{d}{dz}\log(p) =$$

$$\frac{\partial \log(p)}{\partial W_i} =$$

$$\frac{\partial p}{\partial W} =$$

Question: Compute Derivatives

$$\frac{d}{dz}\sigma(z) = \sigma(z) \cdot (1 - \sigma(z))$$

$$p = \sigma(z) = \frac{1}{1+e^{-z}} = \frac{1}{1+e^{-(\mathbf{w}^\top \mathbf{x}+b)}}$$

$$\frac{d}{dp}\log(p) = \frac{1}{p}$$

$$\frac{d}{dp}\log(1-p) = \frac{-1}{1-p}$$

$$\frac{d}{dz}\log(p) = \frac{1}{\sigma(z)} \cdot \sigma(z) \cdot (1 - \sigma(z)) = (1 - \sigma(z))$$

= 1 - p

$$\frac{\partial \log(p)}{\partial W_i} = \frac{1}{p} \cdot \frac{\partial p}{\partial W_i} = \frac{1}{p} \cdot p(1-p) \cdot X_i = (1-p) \cdot X_i$$

$$\frac{\partial p}{\partial W} = p(1-p) \cdot X$$

Logloss Derivative - Reading Homework

$$
\begin{aligned}
\frac{\partial}{\partial \mathbf{w}} \text{LogLoss} &= \frac{\partial}{\partial \mathbf{w}} \left[-y \log(p) - (1-y) \log(1-p) \right] \\
&= -y \frac{\partial}{\partial \mathbf{w}} \log(p) - (1-y) \frac{\partial}{\partial \mathbf{w}} \log(1-p) \\
&= -y \cdot \frac{1}{p} \cdot \frac{\partial p}{\partial \mathbf{w}} - (1-y) \cdot \frac{1}{1-p} \cdot \left(-\frac{\partial p}{\partial \mathbf{w}} \right) \\
&= (p-y) \cdot \mathbf{x}
\end{aligned}
$$

Note

$$\frac{\partial p}{\partial W} = p(1-p) \cdot X$$

- Read denser derving from [here]
- P = sigmoid results
- Y is ground truth

Now

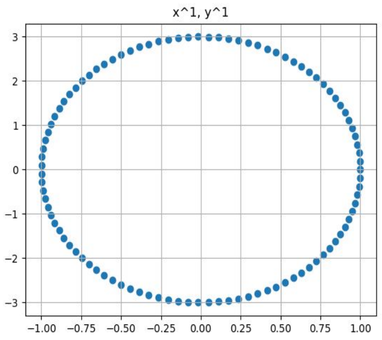
- We introduced a new cost function
- We computed its derivative
    - Which is very similar to linear regression
    - Trivial to implement
- Just pass the computed gradient to the gradient descent
    - Recall: gradient descent is generic: just provide a derivative function
    - Just average the gradient summation of all examples

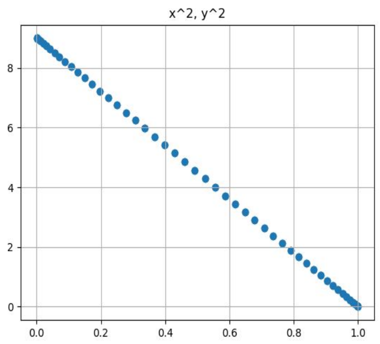
Convex Loss

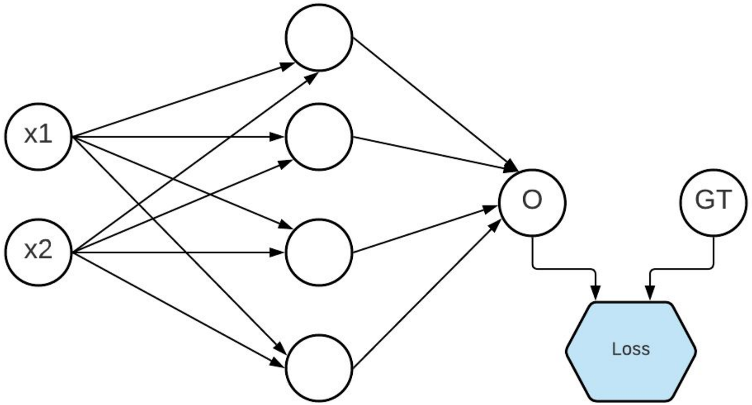
- Is the log loss convex? Surprisingly, the answer is yes!
- To determine convexity, one popular way:
  - The second derivative for **univariate** functions (see), e.g. wx + b   (where x is scalar input)
    - A function is convex if its **2nd derivative** is non-negative everywhere
    - Fair to derive for you
  - The Hessian matrix for **multivariate** functions (see)   e.g. W^TX
    - **Hessian matrix** is a square matrix of **2nd partial derivative** of a scalar-valued function
    - A function is convex if its Hessian matrix is positive semi-definite
    - Requires more math skills

From Logistic Loss to NN

- Moving form logistic loss to NN is as same as moving from linear regression to NN. We just make the transformation non-linear, ending up with logit followed by a sigmoid activation function
- While this is not a convex network with non-linear activations, the transformations can make the a local-minima well trained model better than logistic regression







Summary

- In theory, we can use any valid loss function (there are many)
- However, some of them might have some advantage
  - Convex
  - Better interpretation
  - Faster convergence
  - Less sensitivity for outliers
  - Speed of computations
  - More suitable for specific data assumptions
- At the moment, the logloss is the way-to-go for binary classification
  - We will see its extension for multi-class classification

Relevant Materials

- Animations of Logistic Regression with Python
- Log-odds: link, link, link
- Log-loss: link, Link

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."