Classification using Logistic Regression II

Swati Mishra

Applications of Machine Learning (4AL3)

Fall 2024

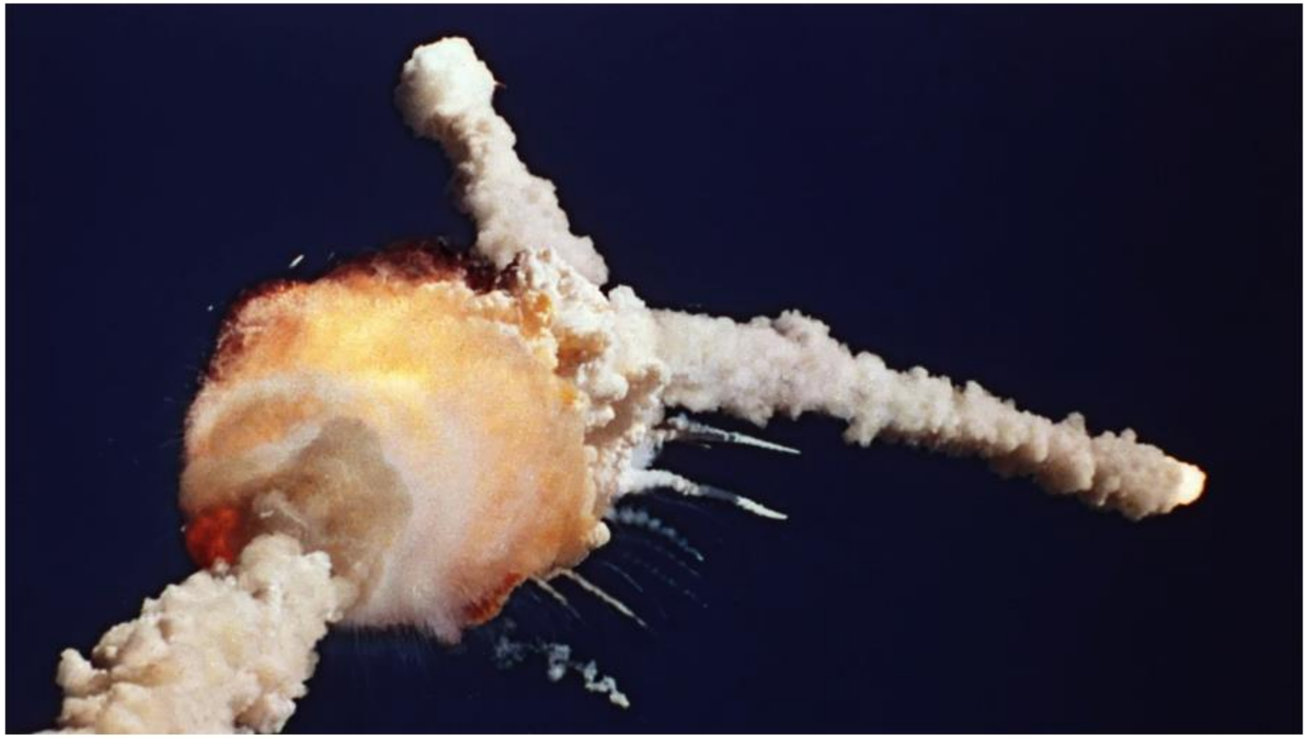
McMaster University
ENGINEERING

Review

- Challenger Disaster
- Classification problems
- Logistic Regression
- Sigmoid Activation Function

Review: Challenger Dataset

Will the O-rings fail catastrophically on the launch day because of the cold weather ?



$$P(y = 1|\, x) \;=\; \sigma(b + \mathbf{w}.\mathbf{x})$$

Learned parameters:

$$b = 10.875 \qquad W = -0.171$$

$$P(y = 1|\, x) = \sigma(10.875 - 0.171x)$$

Probability of failure of O-ring at $\mathrm{x} = 31°\mathrm{F}$

$$P(y = 1) \;=\; \frac{1}{1 + e^{-(10.875 + \mathbf{0.171} * \mathbf{31})}} \;=\; 0.99\%$$

Classification using Logistic Regression

How do we learn $\boldsymbol{W}$ and $\boldsymbol{b}$?

Classification using Logistic Regression

How do we learn $\boldsymbol{W}$ and $\boldsymbol{b}$?

Step 1: We need a loss function

Loss Function

$$P(y_i|x_i) = \begin{cases} p(x_i) & , if\ y_i = 1 \\ 1 - p(x_i) & , if\ y_i = 0 \end{cases}$$

$$p(x) = \sigma(b + \boldsymbol{W}.\boldsymbol{X})$$

Goal is to learn **W** and **b** to maximize the log probability of correct label $p(y|x)$ in training data.

Loss Function

$$P(y_i|x_i) = \begin{cases} p(x_i) & , if\ y_i = 1 \\ 1 - p(x_i) & , if\ y_i = 0 \end{cases}$$

$$p(x) = \sigma(b + \boldsymbol{W}.\boldsymbol{X})$$

Goal is to learn **W** and **b** to maximize the log probability of correct label $p(y|x)$ in training data.

Let's talk about 1 observation x_i,

$$log(p(y_i|x_i)) = y_i log\big(p(x_i)\big) + (1 - y_i)\, log(\, 1 - p(x_i)\,)$$

Loss Function

$$P(y_i|x_i) = \begin{cases} p(x_i) & , if\ y_i = 1 \\ 1 - p(x_i) & , if\ y_i = 0 \end{cases}$$

Let's talk about 1 observation x_i

$$p(x) = \sigma(b + \boldsymbol{W}.\boldsymbol{X})$$

Goal is to learn **W** and **b** to maximize the log probability of correct label $p(y|x)$ in training data.

$$log(p(y_i|x_i)) = y_i log\big(p(x_i)\big) + (1 - y_i)\ log(\ 1 - p(x_i)\ )$$

This function is called the **log likelihood**.

Loss Function

$$P(y_i|x_i) = \begin{cases} p(x_i) & , if\ y_i = 1 \\ 1 - p(x_i) & , if\ y_i = 0 \end{cases}$$

$$p(x) = \sigma(b + \boldsymbol{W}.\boldsymbol{X})$$

Let's talk about 1 observation x_i

Goal is to learn **W** and **b** to **maximize** the log probability of correct label $p(y|x)$ in training data.

$$log(p(y_i|x_i)) = y_i log\big(p(x_i)\big) + (1 - y_i)\ log(\ 1 - p(x_i)\ )$$

This function is called the **log likelihood**.

Loss Function: Cross Entropy Loss

$$P(y_i|x_i) = \begin{cases} p(x_i) & ,if\ y_i = 1 \\ 1 - p(x_i) & ,if\ y_i = 0 \end{cases}$$

$$p(x) = \sigma(b + \boldsymbol{W}.\boldsymbol{X})$$

Goal is to learn **W** and **b** to maximize the log probability of correct label $p(y|x)$ in training data.

Let's talk about 1 observation x_i

$$log(p(y_i|x_i)) = -(y_i log(p(x_i)) + (1 - y_i)\ log(1 - p(x_i)))$$

Loss function needs to be **minimized**. This function is called the **cross-entropy loss** or **negative log likelihood**.

Loss Function: Cross Entropy Loss

$$P(y_i|x_i) = \begin{cases} p(x_i) & ,if\ y_i = 1 \\ 1 - p(x_i) & ,if\ y_i = 0 \end{cases}$$

$$p(x) = \sigma(b + \boldsymbol{W}.\boldsymbol{X})$$

Goal is to learn **W** and **b** to maximize the log probability of correct label $p(y|x)$ in training data.

Let's talk about 1 observation x_i

$$log(p(y_i|x_i)) = -(y_i log(p(x_i)) + (1 - y_i)\ log(\ 1 - p(x_i)\ ))$$

Loss function needs to be minimized. This function is called the **cross-entropy loss** or **negative log likelihood**.

How does this work for our example ?

Classification using Logistic Regression

Cross Entropy Loss after replacing $p(x) = \sigma(b + \boldsymbol{W}.\boldsymbol{X})$

$$log(p(y_i|x_i)) = -(y_i log(\sigma(b + \boldsymbol{w}.\boldsymbol{x})) + (1 - y_i)\, log(1 - \sigma(b + \boldsymbol{w}.\boldsymbol{x})))$$

Learned Parameters

$$b = 10.875$$

$$W = -0.171$$

Classification using Logistic Regression

Cross Entropy Loss after replacing $p(x) = \sigma(b + \boldsymbol{W}.\boldsymbol{X})$

$$log(p(y_i|x_i)) = -(y_i log(\sigma(b + \boldsymbol{w}.\boldsymbol{x})) + (1 - y_i)\, log(1 - \sigma(b + \boldsymbol{w}.\boldsymbol{x})))$$

Learned Parameters

$$b = 10.875$$
$$W = -0.171$$

Let's say the correct label for $x = 31$ is $y = 1$, then Cross Entropy loss = $-\log(\sigma(10.875 - 0.171 * 31))$

Classification using Logistic Regression

Cross Entropy Loss after replacing $p(x) = \sigma(b + \boldsymbol{W}.\boldsymbol{X})$

$$log(p(y_i|x_i)) = -(y_i log(\sigma(b + \boldsymbol{w}.\boldsymbol{x})) + (1 - y_i)\ log(\ 1 - \sigma(b + \boldsymbol{w}.\boldsymbol{x})))$$

Learned Parameters

$b = 10.875$

$W = -0.171$

Let's say the correct label for $x = 31$ is $y = 1$, then Cross Entropy loss = $-\log(\sigma(10.875 - 0.171 * 31))$

Let's say the correct label for $x = 31$ is $y = 0$, then Cross Entropy loss = $-\log(1 - \sigma(10.875 - 0.171 * 31))$

Classification using Logistic Regression

Cross Entropy Loss after replacing $p(x) = \sigma(b + \boldsymbol{W}.\boldsymbol{X})$

$$log(p(y_i|x_i)) = -(y_i log(\sigma(b + \boldsymbol{w}.\boldsymbol{x})) + (1 - y_i)\, log(\, 1 - \sigma(b + \boldsymbol{w}.\boldsymbol{x})))$$

Learned Parameters

$$b = 10.875$$
$$W = -0.171$$

Let’s say the correct label for $x = 31$ is $y = 1$, then Cross Entropy loss = $\log(\sigma(10.875 - 0.171 * 31))$

Let’s say the correct label for $x = 31$ is $y = 0$, then Cross Entropy loss = $-\log(1 - \sigma(10.875 - 0.171 * 31))$

CE Loss ensures probability of the correct answer is maximized

Classification using Logistic Regression

Cross Entropy Loss after replacing $p(x) = \sigma(b + \boldsymbol{W}.\boldsymbol{X})$

$$log(p(y_i|x_i)) = -(y_i log(\sigma(b + \boldsymbol{w}.\boldsymbol{x})) + (1 - y_i)\, log(1 - \sigma(b + \boldsymbol{w}.\boldsymbol{x})))$$

Learned Parameters

$b = 10.875$

$W = -0.171$

Let's say the correct label for $x = 31$ is $y = 1$, then Cross Entropy loss = $\log(\sigma(10.875 - 0.171 * 31))$

Let's say the correct label for $x = 31$ is $y = 0$, then Cross Entropy loss = $-\log(1 - \sigma(10.875 - 0.171 * 31))$

CE Loss ensures probability of the incorrect answer is minimized

Classification using Logistic Regression

How do we learn $\boldsymbol{W}$ and $\boldsymbol{b}$?

Step 2: We need an optimization algorithm

Classification using Logistic Regression

- Sigmoid function is differentiable
- Logistic regression is convex
    - At most one minimum
    - No local minimum to get stuck

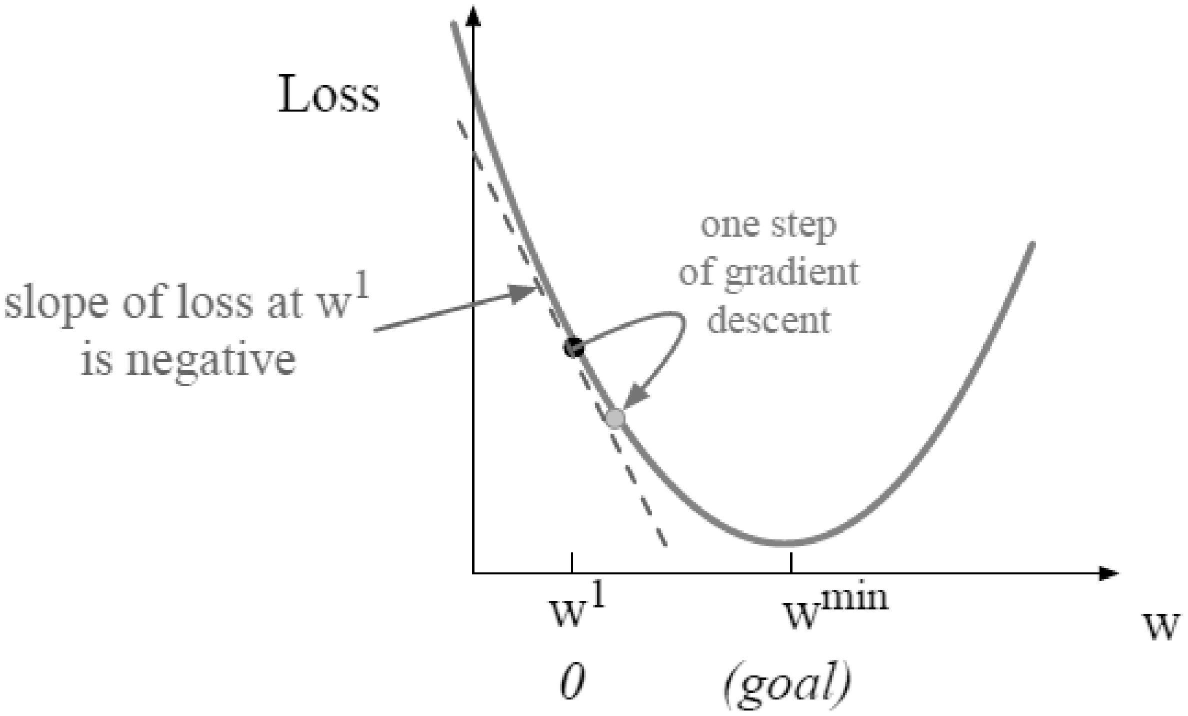
Classification using Logistic Regression

- Sigmoid function is differentiable
- Logistic regression is convex
    - At most one minimum
    - No local minimum to get stuck

Properties of functions that can be minimized using gradient descent

Classification using Logistic Regression

- Sigmoid function is differentiable
- Logistic regression is convex
  - At most one minimum
  - No local minimum to get stuck



Picture Source: *Speech and Language Processing. Daniel Jurafsky & James H. Martin. Copyright © 2023. Draft of February 3, 2024*

Classification using Logistic Regression

- Sigmoid function is differentiable
- Logistic regression is convex
    - At most one minimum
    - No local minimum to get stuck

$$Cross\ Entropy\ Loss(L) = -(y_i log(\sigma(b + \boldsymbol{w}.\boldsymbol{x})) + (1 - y_i)\ log(\ 1 - \sigma(b + \boldsymbol{w}.\boldsymbol{x})))$$

Classification using Logistic Regression

- Sigmoid function is differentiable
- Logistic regression is convex
    - At most one minimum
    - No local minimum to get stuck

$$Cross\ Entropy\ Loss(L) = -(y_i log(\sigma(b + \boldsymbol{w}.\boldsymbol{x})) + (1 - y_i)\ log(\ 1 - \sigma(b + \boldsymbol{w}.\boldsymbol{x})))$$

Gradient of cross entropy loss $\frac{\partial L}{\partial w_i} = [\sigma(b + \boldsymbol{w}.x) - y]x_i = (\mathrm{y}' - \mathrm{y})x_i$

Classification using Logistic Regression

- Sigmoid function is differentiable
- Logistic regression is convex
  - At most one minimum
  - No local minimum to get stuck

$$Cross\ Entropy\ Loss(L) = -(y_i log(\sigma(b + \boldsymbol{w}.\boldsymbol{x})) + (1 - y_i)\ log(\ 1 - \sigma(b + \boldsymbol{w}.\boldsymbol{x})))$$

Gradient of cross entropy loss

$$\frac{\partial L}{\partial w_i} = [\sigma(b + \boldsymbol{w}.x) - y]x_i = (y' - y)x_i$$

$$\frac{\partial L}{\partial b} = [\sigma(b + \boldsymbol{w}.x) - y] = y' - y$$

McMaster University

Classification using Logistic Regression

- Sigmoid function is differentiable
- Logistic regression is convex
  - At most one minimum
  - No local minimum to get stuck

This is good for 1 observation

$$Cross\ Entropy\ Loss(L) = -(y_i log(\sigma(b + \boldsymbol{w}.\boldsymbol{x})) + (1 - y_i)\ log(\ 1 - \sigma(b + \boldsymbol{w}.\boldsymbol{x})))$$

Gradient of cross entropy loss

$$\frac{\partial L}{\partial w_i} = [\sigma(b + \boldsymbol{w}.x) - y]x_i = (y' - y)x_i$$

$$\frac{\partial L}{\partial b} = [\sigma(b + \boldsymbol{w}.x) - y] = y' - y$$

Classification using Logistic Regression

- Sigmoid function is differentiable
- Logistic regression is convex
  - At most one minimum
  - No local minimum to get stuck

What about multiple observations?

$$Cross\ Entropy\ Loss(L) = -(y_i log(\sigma(b + \boldsymbol{w}.\boldsymbol{x})) + (1 - y_i)\ log(\ 1 - \sigma(b + \boldsymbol{w}.\boldsymbol{x})))$$

Gradient of cross entropy loss

$$\frac{\partial L}{\partial w_i} = [\sigma(b + \boldsymbol{w}.x) - y]x_i = (y' - y)x_i$$

$$\frac{\partial L}{\partial b} = [\sigma(b + \boldsymbol{w}.x) - y] = y' - y$$

Classification using Logistic Regression

- Sigmoid function is differentiable
- Logistic regression is convex
  - At most one minimum
  - No local minimum to get stuck

$$Cross\ Entropy\ Loss(L) = -\frac{1}{n}\sum_{i=1}^{n}(y_i log(\sigma(b + \boldsymbol{w}.\boldsymbol{x}))\ +\ (1 - y_i)\ log(\ 1 - \sigma(b + \boldsymbol{w}.\boldsymbol{x})))$$

Classification using Logistic Regression

- Sigmoid function is differentiable
- Logistic regression is convex
  - At most one minimum
  - No local minimum to get stuck

$$Cross\ Entropy\ Loss(L) = -\frac{1}{n}\sum_{i=1}^{n}(y_i log(\sigma(b+\boldsymbol{w}.\boldsymbol{x})) + (1-y_i)\ log(\ 1-\sigma(b+\boldsymbol{w}.\boldsymbol{x})))$$

Gradient of cross entropy loss

$$\frac{\partial L}{\partial w_i} = \frac{1}{n}\ (\sigma(b+\boldsymbol{w}.\boldsymbol{X}) - y)\ X^T$$

$$\frac{\partial L}{\partial b} = \frac{1}{n}\ (\sigma(b+\boldsymbol{w}.\boldsymbol{X}) - y)$$

McMaster University

Classification using Logistic Regression

- Sigmoid function is differentiable
- Logistic regression is convex
  - At most one minimum
  - No local minimum to get stuck

$$Cross\ Entropy\ Loss(L) = -\frac{1}{n}\sum_{i=1}^{n}(y_i log(\sigma(b + \boldsymbol{w}.\boldsymbol{x})) \ + \ (1 - y_i)\ log(\ 1 - \sigma(b + \boldsymbol{w}.\boldsymbol{x})))$$

Gradient of cross entropy loss

$$\frac{\partial L}{\partial w_i} = \frac{1}{n}\ (\overbrace{\sigma(b + \boldsymbol{w}.\boldsymbol{X})}^{\text{Predicted value}} - \underbrace{y}_{\text{Actual value}})\ X^T$$

$$\frac{\partial L}{\partial b} = \frac{1}{n}\ (\sigma(b + \boldsymbol{w}.\boldsymbol{X}) - y)$$

McMaster University

Stochastic Gradient Descent

- Stochastic Gradient Descent is an online algorithm that minimizes the loss function by computing its gradient one sample at a time.

Stochastic Gradient Descent

- Stochastic Gradient Descent is an online algorithm that minimizes the loss function by computing its gradient one sample at a time.

  - Step 1 : Initialize b and w

```
#random uniform distrubution weights
weights=np.random.uniform(size=3)
bias=np.random.uniform(size=3)
```

✓ 0.0s

Stochastic Gradient Descent

- Stochastic Gradient Descent is an online algorithm that minimizes the loss function by computing its gradient one sample at a time.

  - Step 1 : Initialize b and w

  - Step 2: Compute estimated value $y' = \sigma(b + \boldsymbol{w}.\boldsymbol{x})$

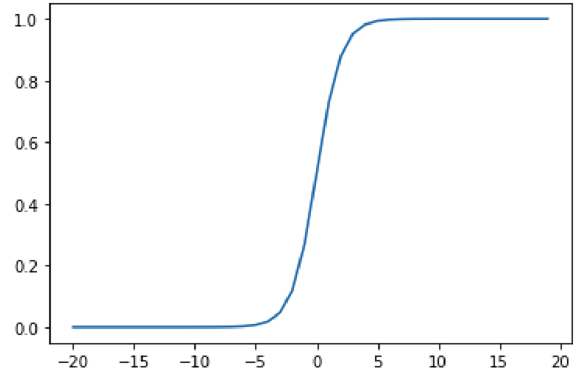
```
def sigmoid( x):
    return 1 / (1 + np.exp(-x))
```

```
y_dash = sigmoid(np.dot(X, weights) + bias)
```

```
x = range(-20,20)
y = [sigmoid(i) for i in range(-20,20)]

plt.plot(x,y)
plt.show()
```

✓ 0.0s



Stochastic Gradient Descent

- Stochastic Gradient Descent is an online algorithm that minimizes the loss function by computing its gradient one sample at a time.

- Step 1 : Initialize b and w
- Step 2: Compute estimated value $y' = \sigma(b + \boldsymbol{w}.\boldsymbol{x})$
- Step 3: Compute the cross-entropy loss $-\frac{1}{n}\sum_{i=1}^{n}(y_i log(\sigma(b + \boldsymbol{w}.\boldsymbol{x})) + (1 - y_i)\, log(1 - \sigma(b + \boldsymbol{w}.\boldsymbol{x})))$

```python
def compute_loss( y_true, y_pred):
        # binary cross entropy
        return -np.mean(y_true * np.log(y_pred ) + (1-y_true) * np.log(1 - y_pred ))
```

Stochastic Gradient Descent

- Stochastic Gradient Descent is an online algorithm that minimizes the loss function by computing its gradient one sample at a time.

- Step 1 : Initialize b and w
- Step 2: Compute estimated value $y' = \sigma(b +$
- Step 3: Compute the cross-entropy loss $-\frac{1}{n}\sum_{i=1}(y_i \log(\sigma(b + w.x)) + (1 - y_i)\log(1 - \sigma(b + w.x)))$
- Step 4: Compute the gradients $\frac{\partial L}{\partial w_i} = \frac{1}{n}\,(\sigma(b + \boldsymbol{w}.\boldsymbol{X}) - y)\,X^T$ $\quad \frac{\partial L}{\partial b} = \frac{1}{n}\,(\sigma(b + \boldsymbol{w}.\boldsymbol{X}) - y)$

```
def compute_gradients(X, y_true, y_pred):
    b =  np.mean(y_pred - y_true)
    # adjust the axis of gradient to compute the mean
    w = np.mean(np.matmul(X.T, y_pred - y_true))
```

Stochastic Gradient Descent

- Stochastic Gradient Descent is an online algorithm that minimizes the loss function by computing its gradient one sample at a time.

- Step 1 : Initialize b and w
- Step 2: Compute estimated value $\mathrm{y}' = \sigma(b + \boldsymbol{w}.\boldsymbol{x})$
- Step 3: Compute the cross-entropy loss $-\frac{1}{n}\sum_{i=1}^{n}(y_i log(\sigma(b + \boldsymbol{w}.\boldsymbol{x})) + (1 - y_i)\, log(\, 1 - \sigma(b + \boldsymbol{w}.\boldsymbol{x})))$
- Step 4: Compute the gradients $\frac{\partial L}{\partial w_i} = \frac{1}{n}\,(\sigma(b + \boldsymbol{w}.\boldsymbol{X}) - y)\, X^T$ $\quad \frac{\partial L}{\partial b} = \frac{1}{n}\,(\sigma(b + \boldsymbol{w}.\boldsymbol{X}) - y)$
- Step 5: Update the parameters $\mathrm{w}' = \mathrm{w} - \alpha\frac{\partial L}{\partial w}$, $\mathrm{b}' = \mathrm{b} - \alpha\frac{\partial L}{\partial b}$

```
w_dash -= alpha * weights
bias_dash -= alpha * bias
```

Classification using Logistic Regression

$$P(y=1|x\ )\ \propto\ \beta_0\ +\sum_{i=1}^{p}\beta_i x_i\ +\ \epsilon$$

We started here on slide 1 previous lecture

To predict after learned weights:

```
def predict(X):
    y_hat = sigmoid(np.dot(X, weights) + bias)
    predicted_class= [1 if i > 0.5 else 0 for i in y_hat]
    return predicted_class
```

We are here now

Classification using Logistic Regression

$$P(y = 1|x) \propto \beta_0 + \sum_{i=1}^{p} \beta_i x_i + \epsilon$$

We started here on slide 1 previous lecture

To predict after learned weights:

```
def predict(X):
    y_hat = sigmoid(np.dot(X, weights) + bias)
    predicted_class= [1 if i > 0.5 else 0 for i in y_hat]
    return predicted_class
```

We are here now

Where did the error term go?

Classification using Logistic Regression

$$P(y = 1|x\,) \propto \beta_0 + \sum_{i=1}^{p} \beta_i x_i + \epsilon$$

Accounting for error term in compute loss

```
def compute_loss( y_true, y_pred):
        # binary cross entropy
        y_pred_errr = y_pred + np.finfo(np.float32).eps
        return -np.mean(y_true * np.log(y_pred_errr ) + (1-y_true) * np.log(1 - y_pred_errr ))
```

Stochastic Gradient Descent

- Stochastic Gradient Descent is an online algorithm that minimizes the loss function by computing its gradient one sample at a time.

- Batch training: Instead of computing one instance at a time, we can compute on the entire dataset

- Mini – batch training: Train on a small subset of the whole dataset:
  - Mini-batches can be easily vectorized
  - Very good for GPU parallelization.

Challenger Dataset

Launch Temp (F)	Did 0-ring get damaged
66	0
70	1
69	0
68	0
67	0
72	0
73	0
70	0
57	1
63	1
70	1
78	0
67	0
53	1
67	0
75	0
70	0
81	0
76	0
79	0
75	1
76	0
58	1

Try running a manual computation on this

Readings

***Required Readings*:**

Introduction to Statistical Learning

1. Chapter 4 – Section 4.1 – 4.3 Page 135 - 144

Supplemental Readings (Not required but recommended):

Deep Learning

1. Chapter 5 – Section 5.5 Page 130-135
2. Chapter 5 – Section 5.9 Page 151-154

Thank You