Deep Learning

Part3: Classification

Ali El Hadi Ismail Fawaz

ENSISA 2A Info, Université Haute-Alsace

November 14, 2024

Une école d'ingénieurs de l'Université de Haute-Alsace

Table of Contents

Binary Classification

Binary Classification

There exists two types of machine learning

- Unsupervised learning (no existence of classes for each example)
- Supervised learning (association of a class label for each example)

Two known tasks on supervised learning

- Regression (predict a continuous value like temperature, pressure etc.)
- Classification (predict a discrete value, class label, like categorizing cats and dogs images etc.)

Binary Classification

There exists two types of machine learning

- Unsupervised learning (no existence of classes for each example)
- Supervised learning (association of a class label for each example)

Two known tasks on supervised learning

- Regression (predict a continuous value like temperature, pressure etc.)
- Classification (predict a discrete value, class label, like categorizing cats and dogs images etc.)

Binary Classification

Dataset

Attack, Defense, Weight and Speed → **Which type the Pokemon belongs to i.e. Water, Fire etc.**

Binary Classification

Dataset

Attack, Defense, Weight and Speed → **Which type the Pokemon belongs to i.e. Water, Fire etc.**

name	weight_kg	speed	sp_attack	sp_defense	type
Zubat	7.5	55	30	40	poison
Charmander	8.5	65	60	50	fire
Butterfree	32	70	90	80	bug
Squirtle	9	43	50	64	water

Binary Classification

Dataset

Attack, Defense, Weight and Speed → **Which type the Pokemon belongs to i.e. Water, Fire etc.**

weight_kg	speed	sp_attack	sp_defense	type
7.5	55	30	40	poison
8.5	65	60	50	fire
32	70	90	80	bug
9	43	50	64	water

Binary Classification

The dataset as is:

weight_kg	speed	sp_attack	sp_defense	type
7.5	55	30	40	poison
8.5	65	60	50	fire
32	70	90	80	bug
9	43	50	64	water

Binary Classification

The dataset as is:

weight_kg	speed	sp_attack	sp_defense	type
7.5	55	30	40	poison
8.5	65	60	50	fire
32	70	90	80	bug
9	43	50	64	water

In this section, we are interested in:

weight_kg	speed	sp_attack	sp_defense	type
7.5	55	30	40	poison
8.5	65	60	50	not poison
32	70	90	80	not poison
9	43	50	64	not poison

Binary Classification

Data Preprocessing

- Normalize the data so that it would be scaled between 0 and 1

Binary Classification

Data Preprocessing

- Normalize the data so that it would be scaled between 0 and 1
- Transform the categorical labels to numerical
  - "poison" $\implies 0$
  - "not poison" $\implies 1$

Binary Classification

Data Preprocessing

- Normalize the data so that it would be scaled between 0 and 1
- Transform the categorical labels to numerical
  - "poison" $\implies 0$
  - "not poison" $\implies 1$
- Split the dataset into train and test sets

Binary Classification

As usual, it is essential to follow these steps:

1. TODO: Inp Out
2. Define the model (hypothesis)
3. Define the objective (cost function)
4. Minimize the cost function with the Gradient Descent algorithm

Binary Classification

Perceptron solving Binary Classification

- Input: $\mathbf{x} = [x_1, x_2, \ldots, x_d]$
- Labels: y can be -1 or 1 (*required*)
- model: $\hat{y} = sign(\mathbf{w}^T.\mathbf{x} + b)$ where $\mathbf{w} = (w_1, w_2, \ldots, w_d)$ the weights, and b the bias and $sign$ is the activation function such as:

$$sign(x) = \begin{cases} +1 & if \ x > 0 \\ -1 & if \ x \leq 0 \end{cases} \tag{1}$$

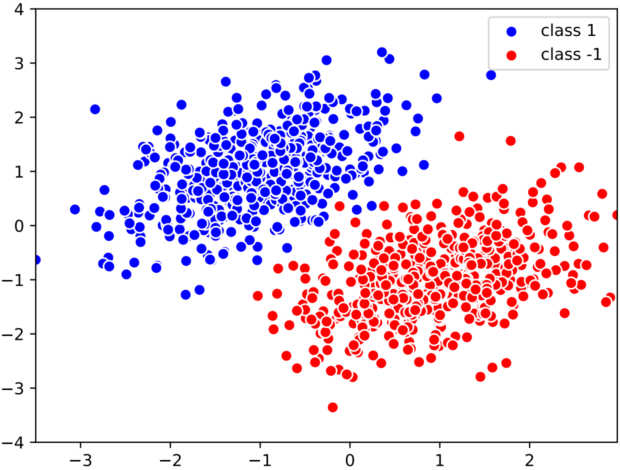
- Loss function is the hinge loss:

$$hinge(y_i, \hat{y}_i) = L(y_i, \hat{y}_i) = \max(0, 1 - y_i.\hat{y}_i) \tag{2}$$

- Gradients are:
  - For all w_i where $i \in \{1, \ldots, d\}$, we have $\frac{\partial L_i}{\partial w_i} = -y_i.x_i$
  - $\frac{\partial L_i}{\partial b} = -y_i$

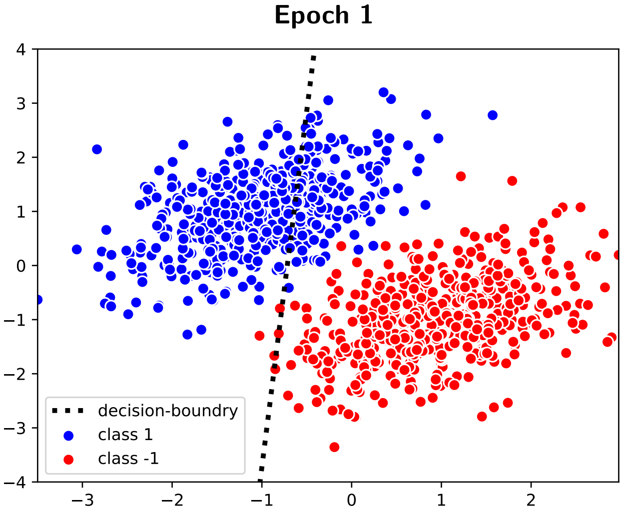
Binary Classification

Perceptron Binary Classifier Decision Boundry



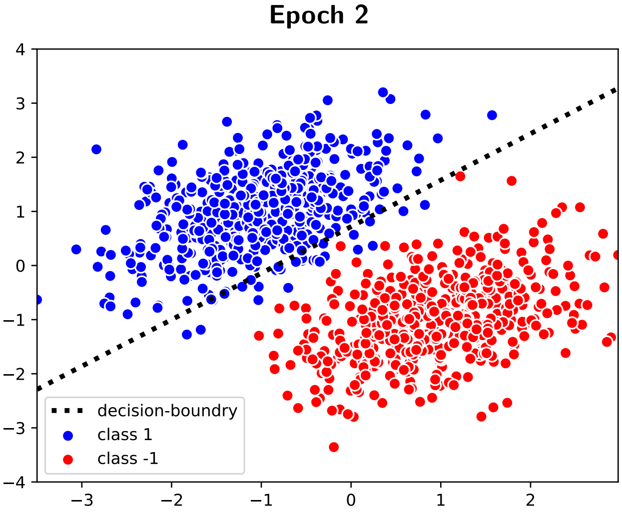
Binary Classification

Perceptron Binary Classifier Decision Boundry



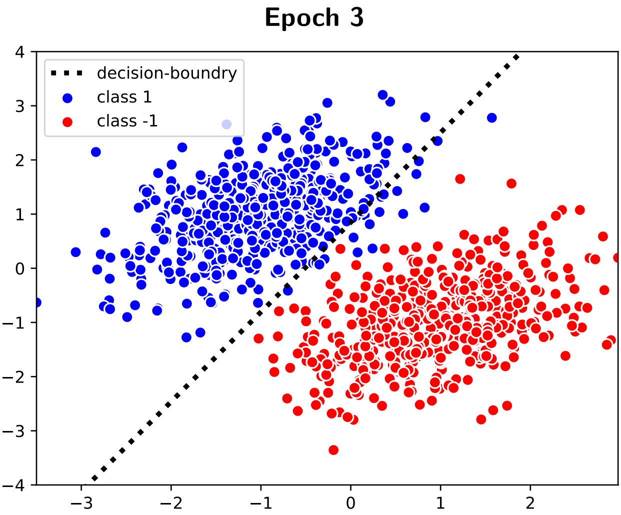
Binary Classification

Perceptron Binary Classifier Decision Boundry



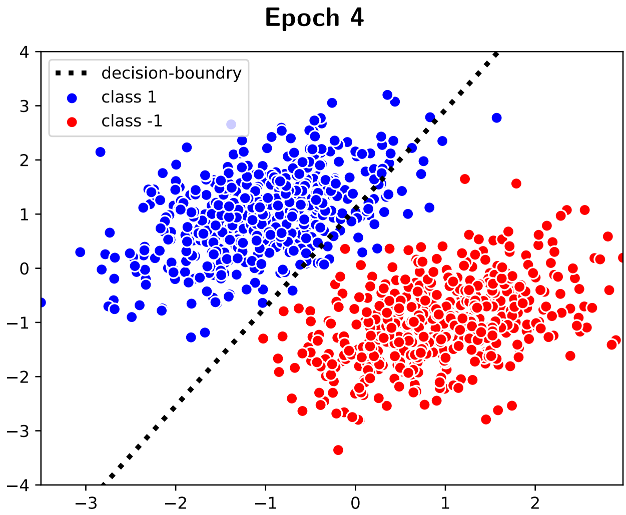
Binary Classification

Perceptron Binary Classifier Decision Boundry



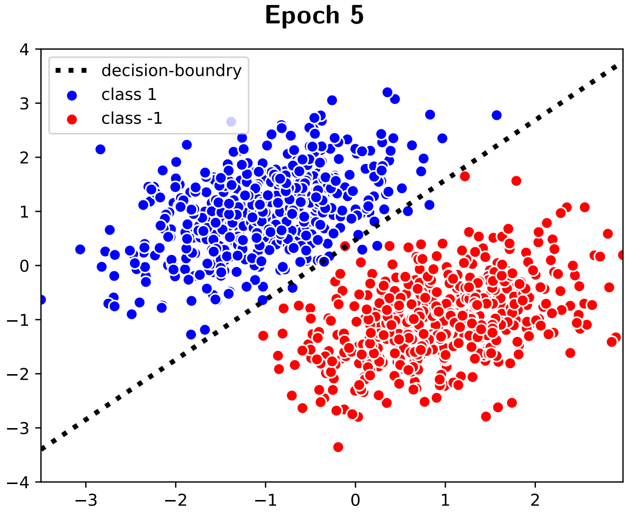
Binary Classification

Perceptron Binary Classifier Decision Boundry



Binary Classification

Perceptron Binary Classifier Decision Boundry



Binary Classification

Limitations of Perceptron Linear Classifier

- Takes long to converge
- Mostly work, and was proposed with Stochastic method, updating on each sample independently
- Does not give a probabilistic view of the predictions, which can be helpful in the case of classification, compared to regression

Binary Classification

Limitations of Perceptron Linear Classifier

- Takes long to converge
- Mostly work, and was proposed with Stochastic method, updating on each sample independently
- Does not give a probabilistic view of the predictions, which can be helpful in the case of classification, compared to regression

Replace with what ?

Binary Classification

Limitations of Perceptron Linear Classifier

- Takes long to converge
- Mostly work, and was proposed with Stochastic method, updating on each sample independently
- Does not give a probabilistic view of the predictions, which can be helpful in the case of classification, compared to regression

Replace with what ?

- Use an activation function that can be interpreted as a probability between 0 and 1
- Example: Sigmoid function

$$sigmoid(x) = \frac{1}{1+e^{-x}} \tag{3}$$

Binary Classification

Limitations of Perceptron Linear Classifier

- Takes long to converge
- Mostly work, and was proposed with Stochastic method, updating on each sample independently
- Does not give a probabilistic view of the predictions, which can be helpful in the case of classification, compared to regression

Replace with what ?

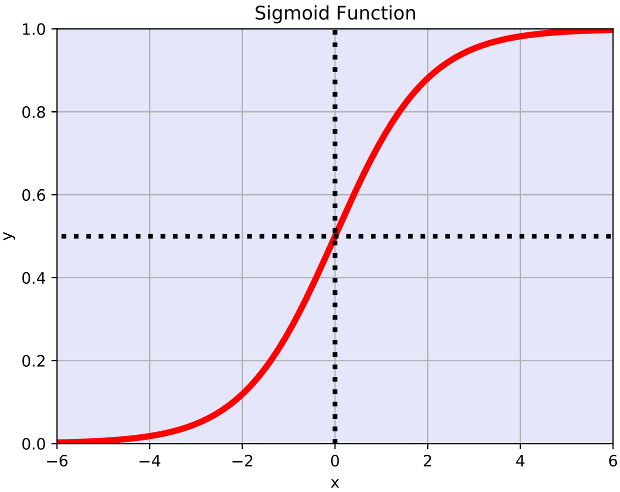
- Use an activation function that can be interpreted as a probability between 0 and 1
- Example: Sigmoid function

$$sigmoid(x) = \frac{1}{1+e^{-x}} \tag{3}$$

- **This is known as Logistic Regression**

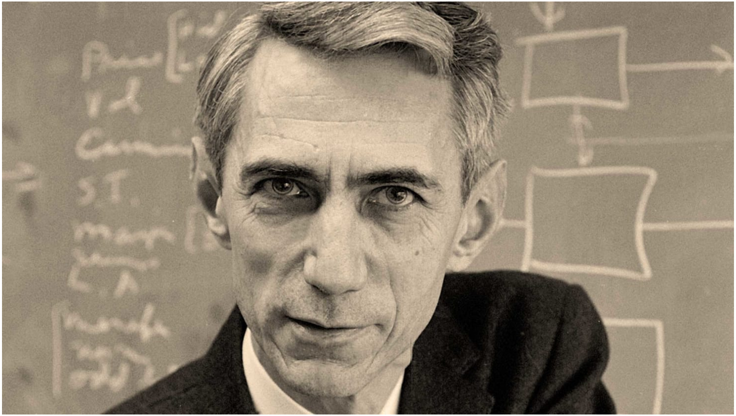
Binary Classification

The sigmoid function



Binary Classification

Defining certitude of a probabilistic event: Claude Shannon introduced the idea of Entropy in 1948



Alfred Eisenstaedt/The LIFE Picture Collection/Getty

Binary Classification

To understand Entropy, take Togepi as an example:

Binary Classification

To understand Entropy, take Togepi as an example:

Binary Classification

To understand Entropy, take Togepi as an example:

Togepi doing the Metronome move

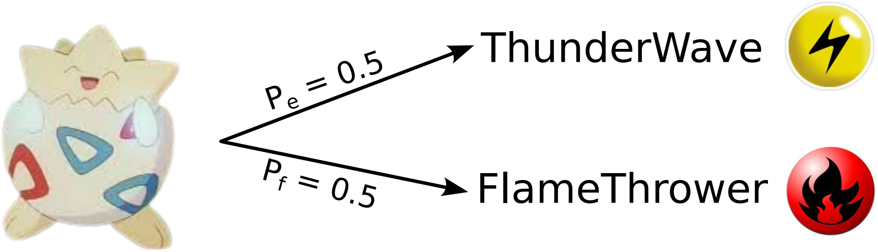
ThunderWave 

FlameThrower 

Binary Classification

To understand Entropy, take Togepi as an example:

Togepi doing the Metronome move

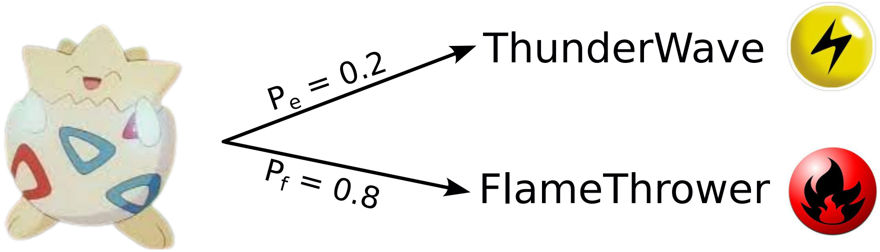


Suppose we don't know the probability of each power being used, and that its as random as a coin flip (50% chance each). The entropy in this case is maximum (maximum uncertainty)

Binary Classification

To understand Entropy, take Togepi as an example:

Togepi doing the Metronome move



Suppose we do know the exact probability of each power being used, the entropy is not maximum, and we are more certain that the power used will be FlameThrower and less certain it will be ThunderWave

Binary Classification

How to calculate the entropy of 2 events belonging to distribution $\mathcal{P}$?

Binary Classification

How to calculate the entropy of 2 events belonging to distribution $\mathcal{P}$?

- For event 1 (power $=$ ThunderWave), the probability is p
- For event 2 (power $=$ FlameThrower), the probability is $1 - p$

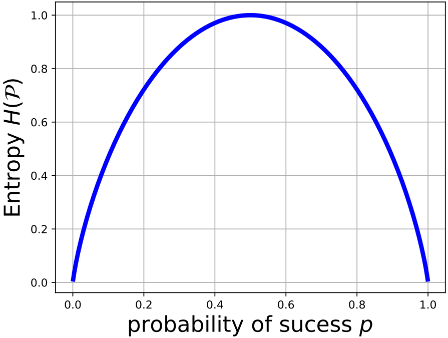
Binary Classification

How to calculate the entropy of 2 events belonging to distribution $\mathcal{P}$?

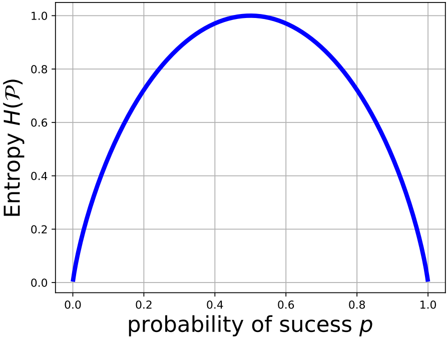
- For event 1 (power = ThunderWave), the probability is p
- For event 2 (power = FlameThrower), the probability is $1-p$

$$H(\mathcal{P}) = -p.\log_2(p) - (1-p).\log_2(1-p) \tag{4}$$

Binary Classification



Binary Classification



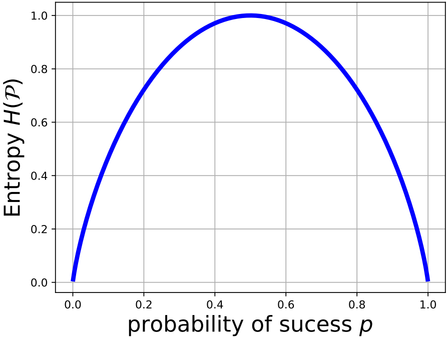
For the case of equiprobable events (50% chance each):

$$H(\mathcal{P}) = -0.5.\log_2(0.5) - 0.5.\log_2(0.5) = 1.0 \tag{5}$$

For the case of un-equiprobable events:

$$H(\mathcal{P}) = -0.2.\log_2(0.2) - 0.8.\log_2(0.8) \approx 0.72193 \tag{6}$$

Binary Classification



General form of entropy for any distribution $\mathcal{P}$ with C events

$$H(\mathcal{P}) = -\sum_{c=1}^{C} p_c . \log_2(p_c) \quad (5)$$

Binary Classification

General form of cross-entropy when finding certitude of cross-distribution $\mathcal{P}$ and $\mathcal{Q}$ with C events

$$CE(\mathcal{P}, \mathcal{Q}) = -\sum_{c=1}^{C} p_c . \log_2(q_c) \tag{6}$$

Binary Classification

General form of cross-entropy when finding certitude of cross-distribution $\mathcal{P}$ and $\mathcal{Q}$ with C events

$$CE(\mathcal{P}, \mathcal{Q}) = -\sum_{c=1}^{C} p_c . \log_2(q_c) \tag{6}$$

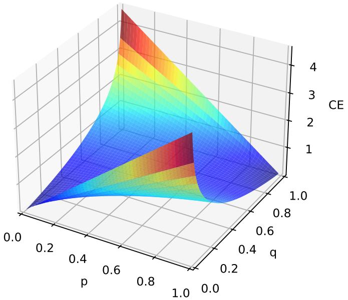
- In the case of two Bernoulli distributions $\mathcal{P}$ and $\mathcal{Q}$:

Binary Classification

General form of cross-entropy when finding certitude of cross-distribution $\mathcal{P}$ and $\mathcal{Q}$ with C events

$$CE(\mathcal{P}, \mathcal{Q}) = -\sum_{c=1}^{C} p_c . \log_2(q_c) \tag{6}$$

- In the case of two Bernoulli distributions $\mathcal{P}$ and $\mathcal{Q}$:



Binary Classification

Logistic Regression

- Input: $\mathbf{x} = [x_1, x_2, \ldots, x_d]$
- Labels: y can be 0 or 1

Binary Classification

Logistic Regression

- Input: $\mathbf{x} = [x_1, x_2, \ldots, x_d]$
- Labels: y can be 0 or 1
- Model: $\hat{y} = sigmoid(\mathbf{w}^T.\mathbf{x} + b) = \dfrac{1}{1 + e^{\mathbf{w}^T.\mathbf{x}+b}}$ where $\mathbf{w} = [w_1, w_2, \ldots, w_d]$ the weights and b the bias

Binary Classification

Logistic Regression

- Input: $\mathbf{x} = [x_1, x_2, \ldots, x_d]$
- Labels: y can be 0 or 1
- Model: $\hat{y} = sigmoid(\mathbf{w}^T.\mathbf{x} + b) = \dfrac{1}{1 + e^{\mathbf{w}^T.\mathbf{x}+b}}$ where $\mathbf{w} = [w_1, w_2, \ldots, w_d]$ the weights and b the bias
- $\hat{y}$ would represent the probability of a sample $\mathbf{x}$ belonging to class 1 and will always very between 0.0 and 1.0
- y would represent the probability of a sample $\mathbf{x}$ belonging to class 1 and will always be deterministic (ground truth) either 0.0 or 1.0

Binary Classification

Logistic Regression

- Input: $\mathbf{x} = [x_1, x_2, \ldots, x_d]$
- Labels: y can be 0 or 1
- Model: $\hat{y} = sigmoid(\mathbf{w}^T.\mathbf{x} + b) = \frac{1}{1 + e^{\mathbf{w}^T.\mathbf{x}+b}}$ where $\mathbf{w} = [w_1, w_2, \ldots, w_d]$ the weights and b the bias
- $\hat{y}$ would represent the probability of a sample $\mathbf{x}$ belonging to class 1 and will always very between 0.0 and 1.0
- y would represent the probability of a sample $\mathbf{x}$ belonging to class 1 and will always be deterministic (ground truth) either 0.0 or 1.0
- Loss function: **Do we use MSE ?**

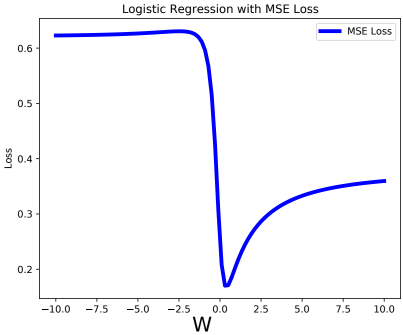
Binary Classification

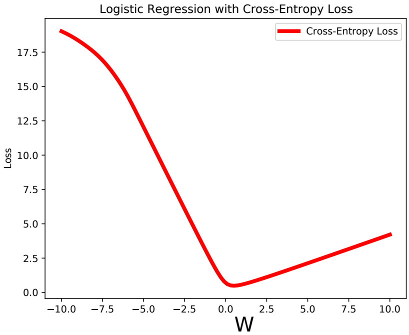
Logistic Regression

- Input: $\mathbf{x} = [x_1, x_2, \ldots, x_d]$
- Labels: y can be 0 or 1
- Model: $\hat{y} = sigmoid(\mathbf{w}^T.\mathbf{x} + b) = \dfrac{1}{1 + e^{\mathbf{w}^T.\mathbf{x}+b}}$ where $\mathbf{w} = [w_1, w_2, \ldots, w_d]$ the weights and b the bias
- $\hat{y}$ would represent the probability of a sample $\mathbf{x}$ belonging to class 1 and will always very between 0.0 and 1.0
- y would represent the probability of a sample $\mathbf{x}$ belonging to class 1 and will always be deterministic (ground truth) either 0.0 or 1.0
- Loss function: **Do we use MSE ?**
- No we use **Cross-Entropy** between the predicted probability and the ground truth

Binary Classification

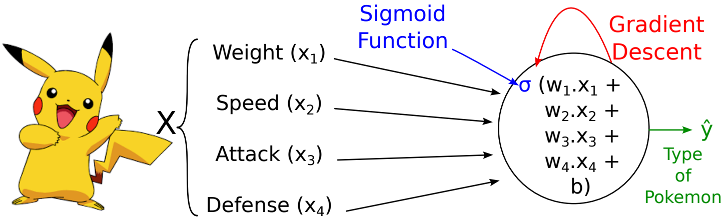
MSE is not convex in Logistic Regression





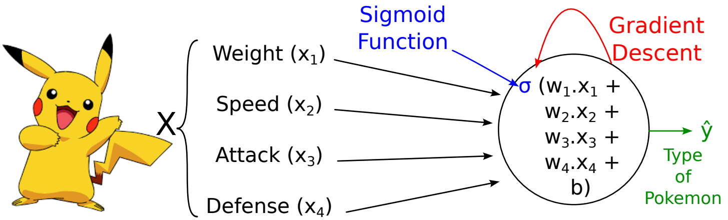
Binary Classification

The Type of Pokemon Classification Task



Binary Classification

The Type of Pokemon Classification Task

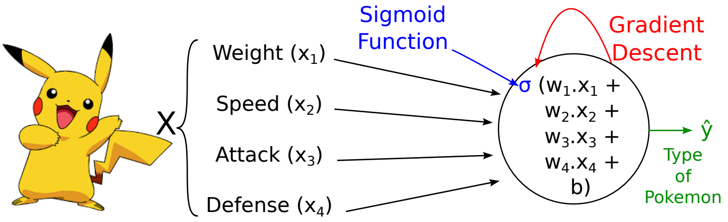


The loss on a batch:

$$L(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} L_i(\mathbf{w}) \tag{7}$$

Binary Classification

The Type of Pokemon Classification Task



The loss on a batch:

$$L(\mathbf{w}) = \frac{1}{n} \sum_{i=1}^{n} L_i(\mathbf{w}) \tag{7}$$

The loss on each sample of the batch

$$L_i(\mathbf{w}) = L_i(y_i, \hat{y}_i) = -y_i . \log_2(\hat{y}_i) - (1 - y_i) . \log_2(1 - \hat{y}_i) \tag{8}$$

Binary Classification

Why use the cross entropy ? (binary cross entropy in this case)

$$L_i(\mathbf{w}) = L_i(y, \hat{y}) = -y_i.\log_2(\hat{y}_i) - (1 - y_i).\log_2(\hat{y}_i) \tag{9}$$

Our goal is to maximize precision. In the Pokemon type binary classification explained before:

Binary Classification

Why use the cross entropy ? (binary cross entropy in this case)

$$L_i(\mathbf{w}) = L_i(y, \hat{y}) = -y_i.\log_2(\hat{y}_i) - (1 - y_i).\log_2(\hat{y}_i) \quad (9)$$

Our goal is to maximize precision. In the Pokemon type binary classification explained before:

- If the ground truth is Poison $y_i = 0$ and the prediction is Poison $\hat{y}_i = 0$

$\Longrightarrow$ The cost $L_i(\mathbf{w})$ should be zero (correct prediction)

$\Longrightarrow$ It is the case: $L_i(\mathbf{w}) = -0.\log_2(0) - (1 - 0).\log_2(1 - 0) = 0$

Binary Classification

Why use the cross entropy ? (binary cross entropy in this case)

$$L_i(\mathbf{w}) = L_i(y, \hat{y}) = -y_i.\log_2(\hat{y}_i) - (1 - y_i).\log_2(\hat{y}_i) \quad (9)$$

Our goal is to maximize precision. In the Pokemon type binary classification explained before:

- If the ground truth is Poison $y_i = 0$ and the prediction is Poison $\hat{y}_i = 0$

  $\Longrightarrow$ The cost $L_i(\mathbf{w})$ should be zero (correct prediction)

  $\Longrightarrow$ It is the case: $L_i(\mathbf{w}) = -0.\log_2(0) - (1 - 0).\log_2(1 - 0) = 0$
- If the ground truth is not-Poison $y_i = 1$ and the prediction is not-Poison $\hat{y}_i = 1$

  $\Longrightarrow$ The cost $L_i(\mathbf{w})$ should be zero (correct prediction)

  $\Longrightarrow$ It is the case: $L_i(\mathbf{w}) = -1.\log_2(1) - (1 - 1).\log_2(1 - 1) = 0$

Binary Classification

Why use the cross entropy ? (binary cross entropy in this case)

$$L_i(\mathbf{w}) = L_i(y, \hat{y}) = -y_i.\log_2(\hat{y}_i) - (1 - y_i).\log_2(\hat{y}_i) \tag{10}$$

Our goal is to maximize precision. In the Pokemon type binary classification explained before:

Binary Classification

Why use the cross entropy ? (binary cross entropy in this case)

$$L_i(\mathbf{w}) = L_i(y, \hat{y}) = -y_i . \log_2(\hat{y}_i) - (1 - y_i) . \log_2(\hat{y}_i) \tag{10}$$

Our goal is to maximize precision. In the Pokemon type binary classification explained before:

- If the ground truth is Poison $y_i = 0$ and the prediction is not-Poison $\hat{y}_i = 1$

$\Longrightarrow$ The cost $L_i(\mathbf{w})$ should be maximum (incorrect prediction)

$\Longrightarrow$ It is the case: $L_i(\mathbf{w}) = -0 . \log_2(1) - (1 - 0) . \log_2(1 - 1) = \infty$

Binary Classification

Why use the cross entropy ? (binary cross entropy in this case)

$$L_i(\mathbf{w}) = L_i(y, \hat{y}) = -y_i.\log_2(\hat{y}_i) - (1 - y_i).\log_2(\hat{y}_i) \tag{10}$$

Our goal is to maximize precision. In the Pokemon type binary classification explained before:

- If the ground truth is Poison $y_i = 0$ and the prediction is not-Poison $\hat{y}_i = 1$

  $\Longrightarrow$ The cost $L_i(\mathbf{w})$ should be maximum (incorrect prediction)

  $\Longrightarrow$ It is the case: $L_i(\mathbf{w}) = -0.\log_2(1) - (1 - 0).\log_2(1 - 1) = \infty$

- If the ground truth is not-Poison $y_i = 1$ and the prediction is Poison $\hat{y}_i = 0$

  $\Longrightarrow$ The cost $L_i(\mathbf{w})$ should be maximum (incorrect prediction)

  $\Longrightarrow$ It is the case: $L_i(\mathbf{w}) = -1.\log_2(0) - (1 - 1).\log_2(1 - 0) = \infty$

Binary Classification

Gradient Descent to solve Logistic Regression

The model: Logistic Regression, we will suppose in what follows that we have one input attribute per sample x and one weight w. We always have one bias b.

$$\hat{y} = \sigma(w.x + b) \tag{11}$$

Binary Classification

Gradient Descent to solve Logistic Regression

The model: Logistic Regression, we will suppose in what follows that we have one input attribute per sample x and one weight w. We always have one bias b.

$$\hat{y} = \sigma(w.x + b) \tag{11}$$

The cost function: Binary Cross Entropy

$$L(w) = L(y, \hat{y}) = -y.\log_2(\hat{y}) - (1-y).\log_2(1-\hat{y}) \tag{12}$$

Binary Classification

Gradient Descent to solve Logistic Regression

The model: Logistic Regression, we will suppose in what follows that we have one input attribute per sample x and one weight w. We always have one bias b.

$$\hat{y} = \sigma(w.x + b) \tag{11}$$

The cost function: Binary Cross Entropy

$$L(w) = L(y, \hat{y}) = -y.\log_2(\hat{y}) - (1-y).\log_2(1-\hat{y}) \tag{12}$$

Optimization Algorithm: Gradient Descent

$$w = w - \alpha.\frac{\partial L}{\partial w} \tag{13}$$

Binary Classification

Gradient Descent to solve Logistic Regression
The model: Logistic Regression, we will suppose in what follows that we have one input attribute per sample x and one weight w. We always have one bias b.

$$\hat{y} = \sigma(w.x + b) \tag{11}$$

The cost function: Binary Cross Entropy

$$L(w) = L(y, \hat{y}) = -y.\log_2(\hat{y}) - (1-y).\log_2(1-\hat{y}) \tag{12}$$

Optimization Algorithm: Gradient Descent

$$w = w - \alpha.\frac{\partial L}{\partial w} \tag{13}$$

$\implies$ The gradient descent method does not change, only the gradient changes $\frac{\partial L}{\partial w}$

Binary Classification

Gradient calculation of the loss function with respect to the parameters.

$$\frac{\partial L(w)}{\partial w} = \frac{\partial[-y\log(\hat{y}) - (1-y)\log(1-\hat{y})]}{\partial w} \tag{14}$$

Binary Classification

Gradient calculation of the loss function with respect to the parameters.

$$\frac{\partial L(w)}{\partial w} = \frac{\partial[-y\log(\hat{y}) - (1-y)\log(1-\hat{y})]}{\partial w} \tag{14}$$

But y **is independent of** $w \implies \frac{\partial y}{\partial w} = 0$

$$\frac{\partial L(w)}{\partial w} = -y\frac{\partial \log(\hat{y})}{\partial w} - (1-y)\frac{\partial \log(1-\hat{y})}{\partial w} \tag{15}$$

Binary Classification

Gradient calculation of the loss function with respect to the parameters.

$$\frac{\partial L(w)}{\partial w} = \frac{\partial[-y\log(\hat{y}) - (1-y)\log(1-\hat{y})]}{\partial w} \tag{14}$$

But y **is independent of** $w \implies \frac{\partial y}{\partial w} = 0$

$$\frac{\partial L(w)}{\partial w} = -y\frac{\partial \log(\hat{y})}{\partial w} - (1-y)\frac{\partial \log(1-\hat{y})}{\partial w} \tag{15}$$

$$\textbf{but} \quad \frac{\partial \log(\hat{y})}{\partial w} = \frac{\frac{\partial \hat{y}}{\partial w}}{\hat{y}} \tag{16}$$

Binary Classification

Gradient calculation of the loss function with respect to the parameters.

$$\frac{\partial L(w)}{\partial w} = \frac{\partial[-y\log(\hat{y}) - (1-y)\log(1-\hat{y})]}{\partial w} \tag{14}$$

But y **is independent of** $w \implies \dfrac{\partial y}{\partial w} = 0$

$$\frac{\partial L(w)}{\partial w} = -y\frac{\partial \log(\hat{y})}{\partial w} - (1-y)\frac{\partial \log(1-\hat{y})}{\partial w} \tag{15}$$

$$\textbf{but} \qquad \frac{\partial \log(\hat{y})}{\partial w} = \frac{\frac{\partial \hat{y}}{\partial w}}{\hat{y}} \tag{16}$$

$$\implies \frac{\partial L(w)}{\partial w} = -y\frac{\frac{\partial \hat{y}}{\partial w}}{\hat{y}} + (1-y)\frac{\frac{\partial \hat{y}}{\partial w}}{1-\hat{y}} \tag{17}$$

Binary Classification

Gradient calculation of the loss function with respect to the parameters.

$$\frac{\partial L(w)}{\partial w} = \frac{\partial[-y\log(\hat{y}) - (1-y)\log(1-\hat{y})]}{\partial w} \tag{14}$$

But y **is independent of** $w \implies \dfrac{\partial y}{\partial w} = 0$

$$\frac{\partial L(w)}{\partial w} = -y\frac{\partial \log(\hat{y})}{\partial w} - (1-y)\frac{\partial \log(1-\hat{y})}{\partial w} \tag{15}$$

$$\textbf{but} \quad \frac{\partial \log(\hat{y})}{\partial w} = \frac{\frac{\partial \hat{y}}{\partial w}}{\hat{y}} \tag{16}$$

$$\implies \frac{\partial L(w)}{\partial w} = -y\frac{\frac{\partial \hat{y}}{\partial w}}{\hat{y}} + (1-y)\frac{\frac{\partial \hat{y}}{\partial w}}{1-\hat{y}} \tag{17}$$

So now we should find $\dfrac{\partial \hat{y}}{\partial w}$

Binary Classification

$$\hat{y} = \sigma(z) \;\;|\;\; z = w.x + b \tag{18}$$

We need to calculate the derivative of $\hat{y}$:

$$\frac{\partial \hat{y}}{\partial w} = \frac{\partial \sigma(z)}{\partial w} \tag{19}$$

Binary Classification

$$\hat{y} = \sigma(z) \ \mid \ z = w.x + b \tag{18}$$

We need to calculate the derivative of $\hat{y}$:

$$\frac{\partial \hat{y}}{\partial w} = \frac{\partial \sigma(z)}{\partial w} \tag{19}$$

$$\textbf{but} \quad \sigma(z) = \frac{1}{1 + e^{-z}} = (1 + e^{-z})^{-1} \tag{20}$$

Binary Classification

$$\hat{y} = \sigma(z) \;\; | \;\; z = w.x + b \tag{18}$$

We need to calculate the derivative of $\hat{y}$:

$$\frac{\partial \hat{y}}{\partial w} = \frac{\partial \sigma(z)}{\partial w} \tag{19}$$

$$\textbf{but} \quad \sigma(z) = \frac{1}{1 + e^{-z}} = (1 + e^{-z})^{-1} \tag{20}$$

$$\Longrightarrow \;\; \frac{\partial \sigma(z)}{\partial w} = (-1)\frac{\partial (1 + e^{-z})}{\partial w}(1 + e^{-z})^{-2} \tag{21}$$

Binary Classification

$$\hat{y} = \sigma(z) \ \ | \ \ z = w.x + b \tag{18}$$

We need to calculate the derivative of $\hat{y}$:

$$\frac{\partial \hat{y}}{\partial w} = \frac{\partial \sigma(z)}{\partial w} \tag{19}$$

$$\textbf{but} \quad \sigma(z) = \frac{1}{1+e^{-z}} = (1+e^{-z})^{-1} \tag{20}$$

$$\Longrightarrow \ \frac{\partial \sigma(z)}{\partial w} = (-1)\frac{\partial(1+e^{-z})}{\partial w}(1+e^{-z})^{-2} \tag{21}$$

$$\textbf{but} \quad \frac{\partial(1+e^{-z})}{\partial w} = \frac{\partial e^{-z}}{\partial w} \tag{22}$$

Binary Classification

$$\hat{y} = \sigma(z) \;\mid\; z = w.x + b \tag{18}$$

We need to calculate the derivative of $\hat{y}$:

$$\frac{\partial \hat{y}}{\partial w} = \frac{\partial \sigma(z)}{\partial w} \tag{19}$$

$$\textbf{but} \quad \sigma(z) = \frac{1}{1+e^{-z}} = (1+e^{-z})^{-1} \tag{20}$$

$$\implies \frac{\partial \sigma(z)}{\partial w} = (-1)\frac{\partial(1+e^{-z})}{\partial w}(1+e^{-z})^{-2} \tag{21}$$

$$\textbf{but} \quad \frac{\partial(1+e^{-z})}{\partial w} = \frac{\partial e^{-z}}{\partial w} \tag{22}$$

So we need to calculate $\dfrac{\partial e^{-z}}{\partial w}$

Binary Classification

We need to calculate

$$\frac{\partial e^{-z}}{\partial w} \tag{23}$$

Binary Classification

We need to calculate

$$\frac{\partial e^{-z}}{\partial w} \tag{23}$$

In accordance with the theorem of the derivative of composite functions

$$\frac{\partial e^{-z}}{\partial w} = \frac{\partial e^{-z}}{\partial z}\frac{\partial z}{\partial w} \tag{24}$$

Binary Classification

We need to calculate

$$\frac{\partial e^{-z}}{\partial w} \tag{23}$$

In accordance with the theorem of the derivative of composite functions

$$\frac{\partial e^{-z}}{\partial w} = \frac{\partial e^{-z}}{\partial z}\frac{\partial z}{\partial w} \tag{24}$$

$$\frac{\partial e^{-z}}{\partial z} = -e^{-z} \tag{25}$$

Binary Classification

We need to calculate

$$\frac{\partial e^{-z}}{\partial w} \tag{23}$$

In accordance with the theorem of the derivative of composite functions

$$\frac{\partial e^{-z}}{\partial w} = \frac{\partial e^{-z}}{\partial z}\frac{\partial z}{\partial w} \tag{24}$$

$$\frac{\partial e^{-z}}{\partial z} = -e^{-z} \tag{25}$$

But $z = w.x + b$

$$\implies \frac{\partial z}{\partial w} = x \tag{26}$$

Binary Classification

We need to calculate

$$\frac{\partial e^{-z}}{\partial w} \tag{23}$$

In accordance with the theorem of the derivative of composite functions

$$\frac{\partial e^{-z}}{\partial w} = \frac{\partial e^{-z}}{\partial z}\frac{\partial z}{\partial w} \tag{24}$$

$$\frac{\partial e^{-z}}{\partial z} = -e^{-z} \tag{25}$$

But $z = w.x + b$

$$\implies \frac{\partial z}{\partial w} = x \tag{26}$$

$$\implies \frac{\partial e^{-z}}{\partial w} = -x.e^{-z} \tag{27}$$

Binary Classification

We recall the original objective

$$\frac{\partial L(w)}{\partial w} = -y\frac{\frac{\partial \hat{y}}{\partial w}}{\hat{y}} + (1-y)\frac{\frac{\partial \hat{y}}{\partial w}}{1-\hat{y}} \tag{28}$$

Binary Classification

We recall the original objective

$$\frac{\partial L(w)}{\partial w} = -y\frac{\frac{\partial \hat{y}}{\partial w}}{\hat{y}} + (1-y)\frac{\frac{\partial \hat{y}}{\partial w}}{1-\hat{y}} \tag{28}$$

By using the equations 21 et 27

$$\frac{\partial \hat{y}}{\partial w} = \frac{\partial \sigma(z)}{\partial w} = \frac{x.e^{-z}}{(1+e^{-z})^2} = x.\sigma(z)(1-\sigma(z)) = x\hat{y}(1-\hat{y}) \tag{29}$$

Binary Classification

We recall the original objective

$$\frac{\partial L(w)}{\partial w} = -y\frac{\frac{\partial \hat{y}}{\partial w}}{\hat{y}} + (1-y)\frac{\frac{\partial \hat{y}}{\partial w}}{1-\hat{y}} \tag{28}$$

By using the equations 21 et 27

$$\frac{\partial \hat{y}}{\partial w} = \frac{\partial \sigma(z)}{\partial w} = \frac{x.e^{-z}}{(1+e^{-z})^2} = x.\sigma(z)(1-\sigma(z)) = x\hat{y}(1-\hat{y}) \tag{29}$$

By replacing (29) in (28) and after simplifications

$$\frac{\partial L(w)}{\partial w} = (\hat{y}-y).x \tag{30}$$

Binary Classification

We recall the original objective

$$\frac{\partial L(w)}{\partial w} = -y\frac{\frac{\partial \hat{y}}{\partial w}}{\hat{y}} + (1-y)\frac{\frac{\partial \hat{y}}{\partial w}}{1-\hat{y}} \tag{28}$$

By using the equations 21 et 27

$$\frac{\partial \hat{y}}{\partial w} = \frac{\partial \sigma(z)}{\partial w} = \frac{x.e^{-z}}{(1+e^{-z})^2} = x.\sigma(z)(1-\sigma(z)) = x\hat{y}(1-\hat{y}) \tag{29}$$

By replacing (29) in (28) and after simplifications

$$\frac{\partial L(w)}{\partial w} = (\hat{y}-y).x \tag{30}$$

In reality we do the average gradient over all samples in a batch of size n

$$\frac{\partial L(w)}{\partial w} = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i).x_i \tag{31}$$

Binary Classification

Now we can update w using the gradient descent method

$$w = w - \alpha.\frac{\partial L(w)}{\partial w} \tag{32}$$

Binary Classification

Now we can update w using the gradient descent method

$$w = w - \alpha.\frac{\partial L(w)}{\partial w} \tag{32}$$

By using 31

$$w = w - \frac{\alpha}{n}.\sum_{i=1}^{n}(\hat{y}_i - y_i).x_i \tag{33}$$

Binary Classification

Now we can update w using the gradient descent method

$$w = w - \alpha.\frac{\partial L(w)}{\partial w} \tag{32}$$

By using 31

$$w = w - \frac{\alpha}{n}.\sum_{i=1}^{n}(\hat{y_i} - y_i).x_i \tag{33}$$

For the bias b, if we follow the same methodology as for w, we end up with:

$$b = b - \frac{\alpha}{n}\frac{\partial L(b)}{\partial b} \tag{34}$$

$$b = b - \frac{\alpha}{n}\sum_{i=1}^{n}(\hat{y_i} - y_i) \tag{35}$$

Binary Classification

Deciding the predicted class after training.

- The output of the neuron is the probability of a sample belonging to class 0 or 1

$$\hat{y}_i = \sigma(w.x_i + b) \tag{36}$$

Binary Classification

Deciding the predicted class after training.

- The output of the neuron is the probability of a sample belonging to class 0 or 1

$$\hat{y}_i = \sigma(w.x_i + b) \tag{36}$$

- To find the predicted class, we create a threshold on 0.5

Binary Classification

Deciding the predicted class after training.

- The output of the neuron is the probability of a sample belonging to class 0 or 1

$$\hat{y}_i = \sigma(w.x_i + b) \tag{36}$$

- To find the predicted class, we create a threshold on 0.5
  - If the probability $\hat{y}_i$ is $> 0.5 \implies$ the predicted class is 1
  - If the probability $\hat{y}_i$ is $\leq 0.5 \implies$ the predicted class is 0

Binary Classification

Evaluation metric

Binary Classification

Evaluation metric

- To evaluate a classification model, we rely on a discrete metric for precision, in this case we will rely on **accuracy metric**

$$accuracy = \frac{\textit{number of correctly predicted samples}}{\textit{total number of samples}} \tag{37}$$

Binary Classification

Evaluation metric

- To evaluate a classification model, we rely on a discrete metric for precision, in this case we will rely on **accuracy metric**

$$accuracy = \frac{\textit{number of correctly predicted samples}}{\textit{total number of samples}} \tag{37}$$

- For example. if we correctly classify the type of 70 Pokemon and we have 100 Pokemon in total:

$$\implies accuracy = \frac{70}{100} = 70\% \tag{38}$$

Binary Classification

Evaluation metric

- To evaluate a classification model, we rely on a discrete metric for precision, in this case we will rely on **accuracy metric**

$$accuracy = \frac{\textit{number of correctly predicted samples}}{\textit{total number of samples}} \tag{37}$$

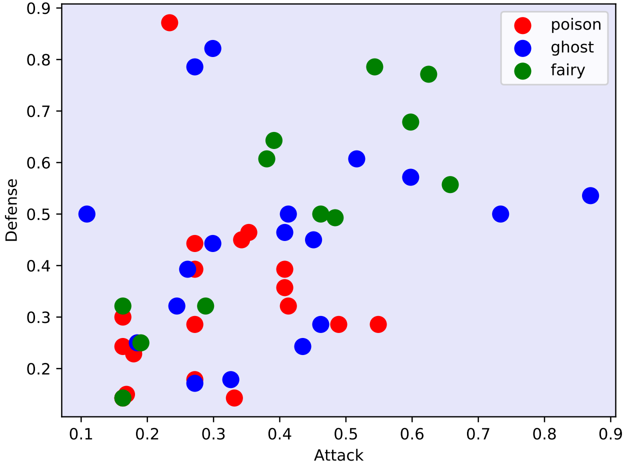
- For example. if we correctly classify the type of 70 Pokemon and we have 100 Pokemon in total:

$$\implies accuracy = \frac{70}{100} = 70\% \tag{38}$$

- This metric is always between 0 and 1 (a percentage)

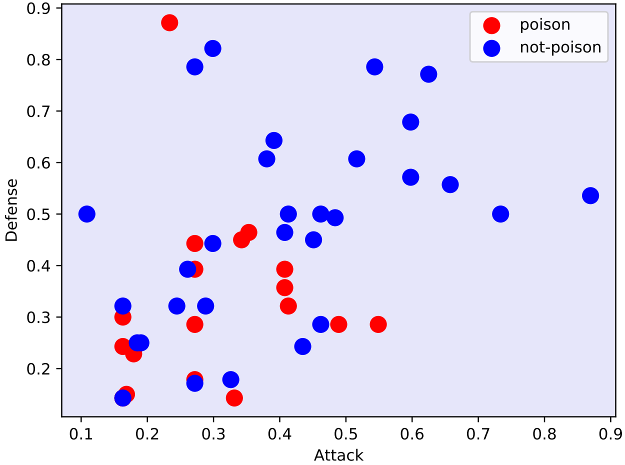
Binary Classification

Predicting Pokemon type using Attack and Defense values



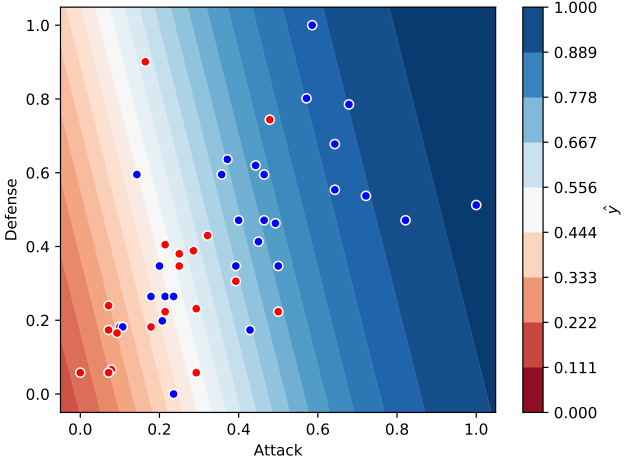
Binary Classification

Predicting Pokemon type using Attack and Defense values



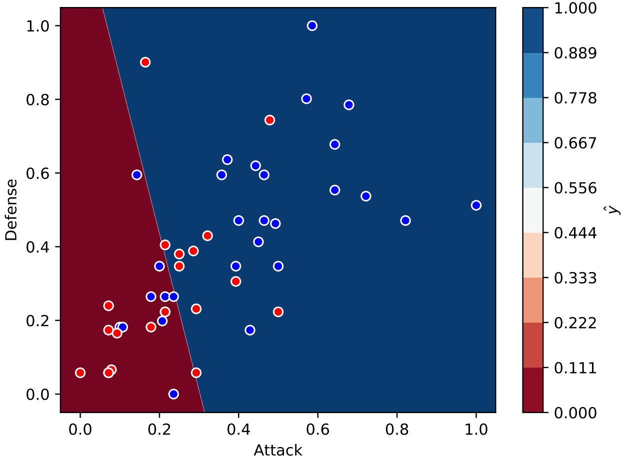
Binary Classification

Predicting Pokemon type - LR decision boundaries - Soft predictions



Binary Classification

Predicting Pokemon type - LR decision boundaries - Hard predictions



Multi-Class Classification

Multi-Class Classification

We will try to predict any of the types, not only poison and not poison

weight_kg	speed	sp_attack	sp_defense	type
7.5	55	30	40	poison
8.5	65	60	50	fire
32	70	90	80	bug
9	43	50	64	water

Multi-Class Classification

Encoding Labels

- Encode labels to numerical values

$$Y = [\text{poison, ghost, fairy, grass, poison,...}] \implies Y = [0, 1, 2, 3, 0, ...]$$

- If the labels are numerical but not continuous

$$Y = [1, 2, 4, 1, ...] \implies [0, 1, 2, 0, ...]$$

Multi-Class Classification

Encoding Labels

- Encode labels to numerical values

$$Y = [\text{poison, ghost, fairy, grass, poison,...}] \implies Y = [0, 1, 2, 3, 0, ...]$$

- If the labels are numerical but not continuous

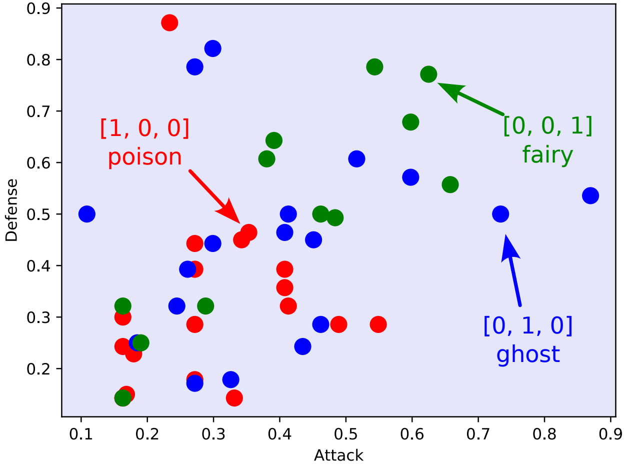
$$Y = [1, 2, 4, 1, ...] \implies [0, 1, 2, 0, ...]$$

One Hot Encoding to make each label a probability distribution

- If we have only three classes $0, 1$ and 2:
- $0 \implies [1, 0, 0]$
- $1 \implies [0, 1, 0]$
- $2 \implies [0, 0, 1]$

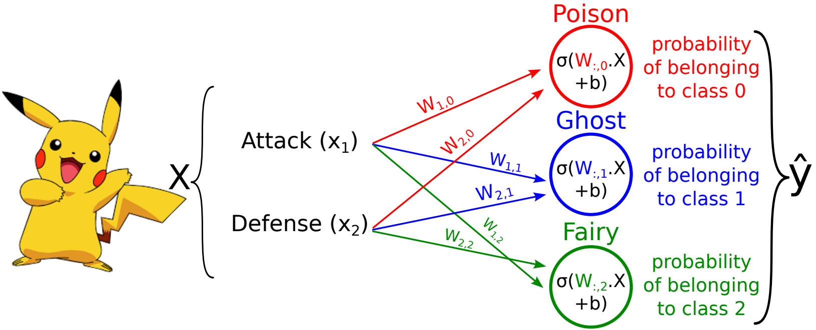
Multi-Class Classification

One Hot Encoding on three classes



Multi-Class Classification

SoftMax Classifier - simple case of two input attributes - same applied for 4 attributes



Multi-Class Classification

- Given a Pokemon with its four attributes:

$$X_i = [X_{i,1}, X_{i,2}, X_{i,3}, X_{i,4}] \tag{39}$$

- The goal is to calculate the probability vector:

$$\hat{Y}_i = [\hat{y}_{i,0}, \hat{y}_{i,1}, \hat{y}_{i,2}] \tag{40}$$

- With:
  - $\hat{y}_{i,0}$ is the probability that X_i belongs to class 0
  - $\hat{y}_{i,1}$ is the probability that X_i belongs to class 1
  - $\hat{y}_{i,2}$ is the probability that X_i belongs to class 2

Multi-Class Classification

Generalizing the Sigmoid function to any number of classes

- $\hat{Y}_i$ is a vector with its elements representing a probability distribution over C classes

$$\hat{Y}_i = [\hat{y}_{i,0}, \hat{y}_{i,1}, \ldots, \hat{y}_{i,C-1}] \tag{41}$$

Multi-Class Classification

Generalizing the Sigmoid function to any number of classes

- $\hat{Y}_i$ is a vector with its elements representing a probability distribution over C classes

$$\hat{Y}_i = [\hat{y}_{i,0}, \hat{y}_{i,1}, \ldots, \hat{y}_{i,C-1}] \tag{41}$$

- This means, every element should follow the constraint:

$$0 \ \leq \ \hat{y}_{i,c} \ \leq \ 1.0 \mid \forall \ j \ \in \ \{0, 1, \ldots, C-1\} \tag{42}$$

Multi-Class Classification

Generalizing the Sigmoid function to any number of classes

- $\hat{Y}_i$ is a vector with its elements representing a probability distribution over C classes

$$\hat{Y}_i = [\hat{y}_{i,0}, \hat{y}_{i,1}, \ldots, \hat{y}_{i,C-1}] \tag{41}$$

- This means, every element should follow the constraint:

$$0 \ \leq \ \hat{y}_{i,c} \ \leq \ 1.0 \mid \forall \ j \ \in \ \{0, 1, \ldots, C-1\} \tag{42}$$

- The sum of elements in $\hat{Y}_i$ should follow the constraint:

$$\sum_{c=0}^{C-1} \hat{y}_{i,c} = 1.0 \tag{43}$$

Multi-Class Classification

Generalizing the Sigmoid function to any number of classes

- $\hat{Y}_i$ is a vector with its elements representing a probability distribution over C classes

$$\hat{Y}_i = [\hat{y}_{i,0}, \hat{y}_{i,1}, \ldots, \hat{y}_{i,C-1}] \tag{41}$$

- This means, every element should follow the constraint:

$$0 \ \leq \ \hat{y}_{i,c} \ \leq \ 1.0 \mid \forall \ j \ \in \ \{0, 1, \ldots, C-1\} \tag{42}$$

- The sum of elements in $\hat{Y}_i$ should follow the constraint:

$$\sum_{c=0}^{C-1} \hat{y}_{i,c} = 1.0 \tag{43}$$

- For this reason we use the softmax function on $z_c = W_{:,c}.X + b$

$$\hat{y}_{i,c} = \frac{e^{z_c}}{\sum_{c=0}^{C-1} e^{z_c}} \tag{44}$$

Multi-Class Classification

For Evaluation

- We choose the winning class for the prediction, following the class with the highest probability:

$$prediction_i = arg\max_c \hat{Y}_i \quad (45)$$

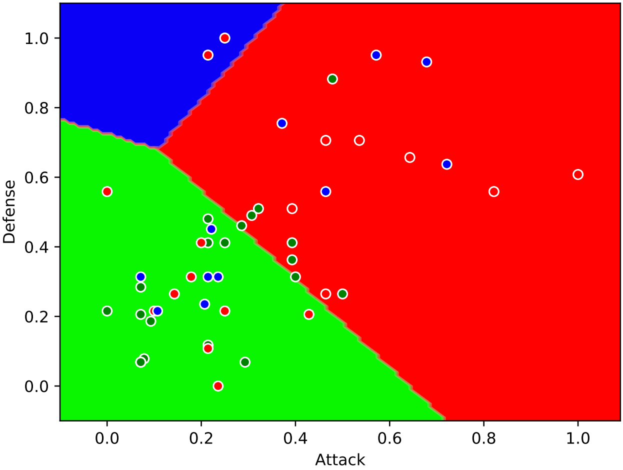
- Which comes down to:

$$prediction_i = arg\max_c(\hat{y}_{i,0}, \hat{y}_{i,1}, \ldots, \hat{y}_{i,C-1}) \quad (46)$$

- The accuracy metric does not change, same as for Binary Classification

Multi-Class Classification

Decision Boundary of Softmax Classifier - Hard Decisions



Multi-Class Classification

See Winning Class Probability - Soft Decisions