Machine Learning I

Mohamed Hussien

Logistic Regression

Lecture Overview

- Throwback
- Linear Regression As A Classifier!
- Problem Definition
- Perceptron Classifier [Formula]
- Perceptron Classifier [error function]
- Perceptron Classifier [Training]
- Perceptron Classifier [SKlearn]
- Logistic regression vs Perceptron Classifier
- Logistic regression [cross entropy error]
- Logistic regression [Training]
- Logistic regression [SKlearn]
- Logistic regression [Multi-class Model]

Classification vs Regression

Classification Models

> The types of models that predict categorical data. The output of a classification model is a category, or a state, such as the type of animal (cat or dog).

Example: Email spam detection model

Regression Models

> The types of models that predict numerical data. The output of a regression model is a number, such as the weight of the animal.

Example: Housing prices model

Classification vs Regression

Labelled Data

Categorical Data

Labelled Data

Numerical Data

Lecture Overview

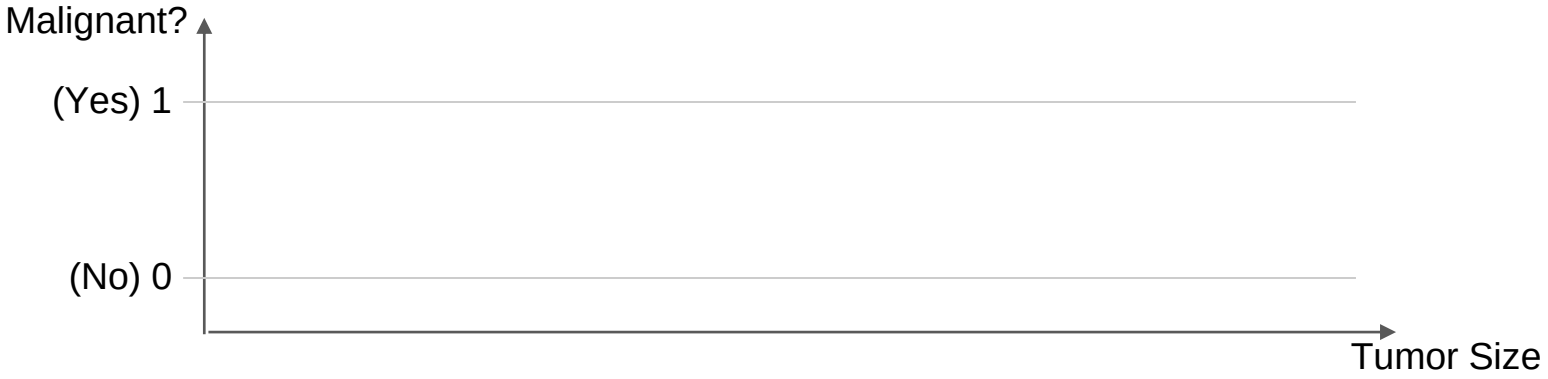
- **Throwback**
- **Linear Regression As A Classifier!**
- **Problem Definition**
- **Perceptron Classifier [Formula]**
- **Perceptron Classifier [error function]**
- **Perceptron Classifier [Training]**
- **Perceptron Classifier [SKlearn]**
- **Logistic regression vs Perceptron Classifier**
- **Logistic regression [cross entropy error]**
- **Logistic regression [Training]**
- **Logistic regression [SKlearn]**
- **Logistic regression [Multi-class Model]**

Linear Regression As A Classifier!

Why not use linear regression as a classifier?

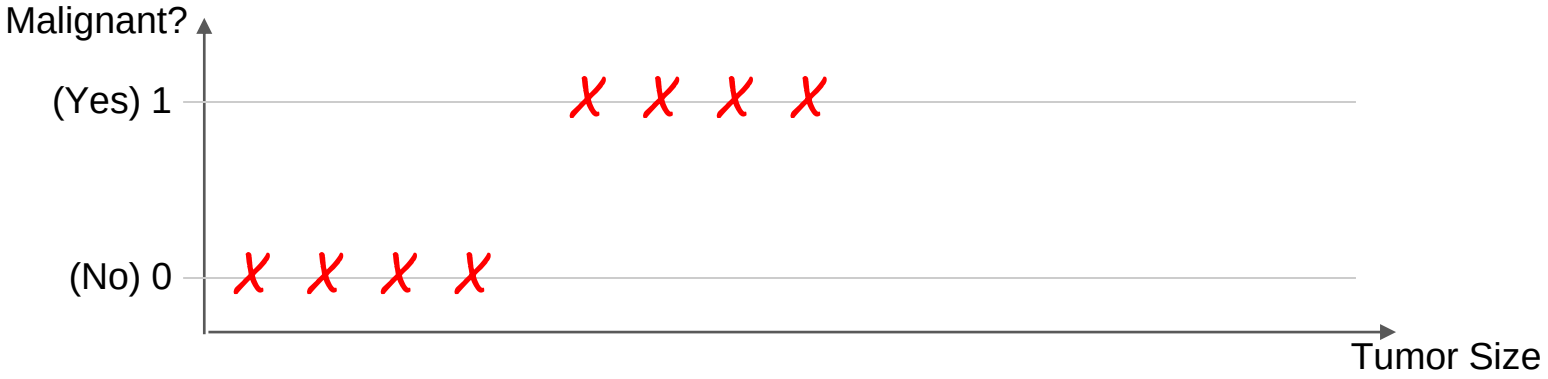
Linear Regression As A Classifier!

Why not use linear regression as a classifier?



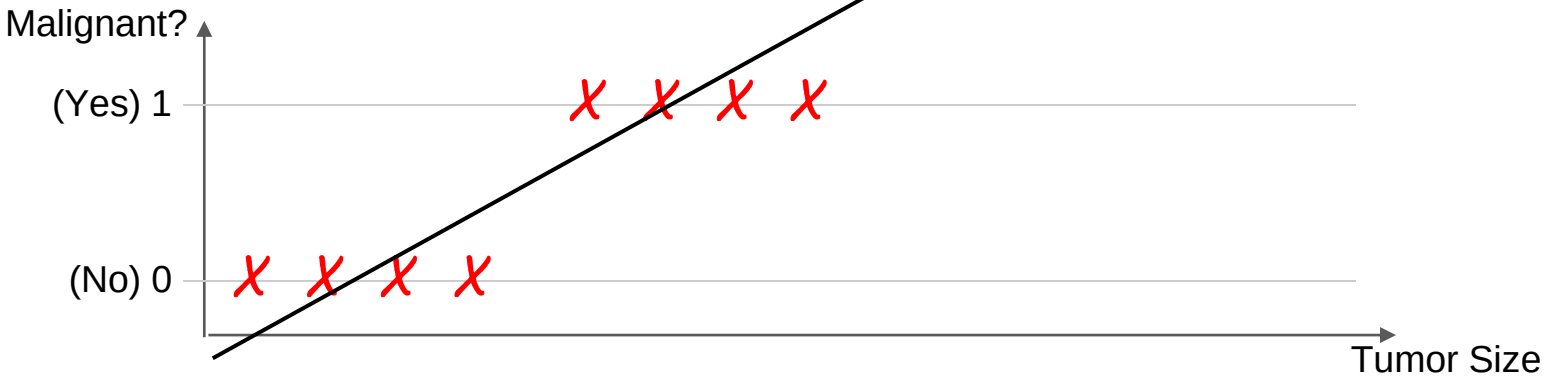
Linear Regression As A Classifier!

Why not use linear regression as a classifier?



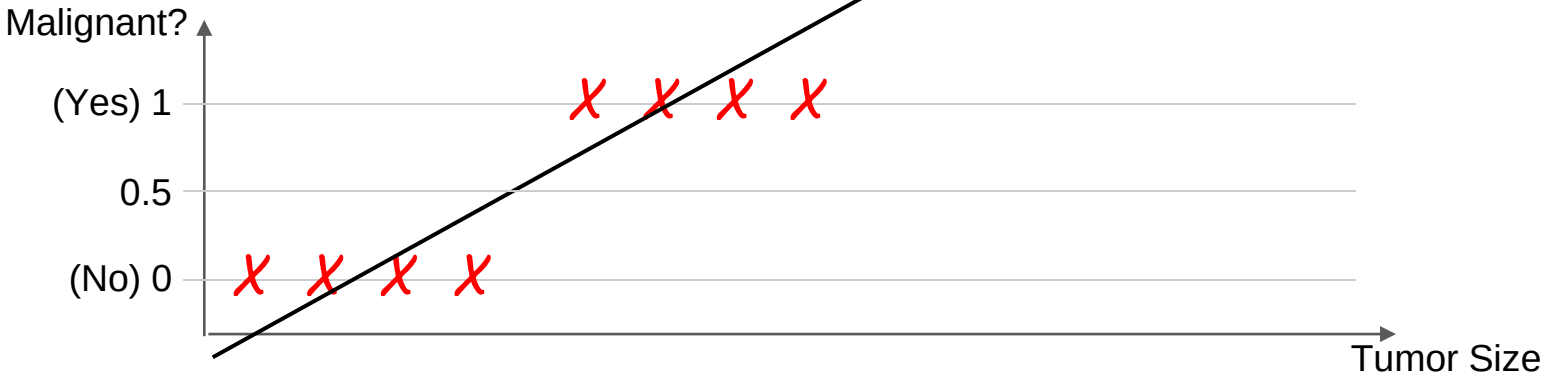
Linear Regression As A Classifier!

Why not use linear regression as a classifier?



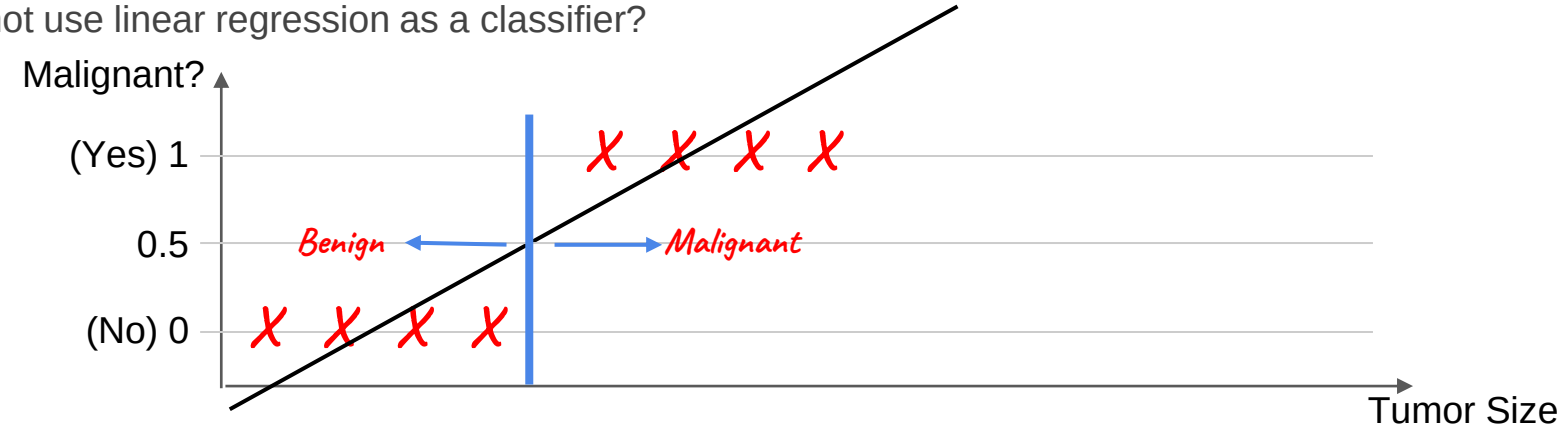
Linear Regression As A Classifier!

Why not use linear regression as a classifier?



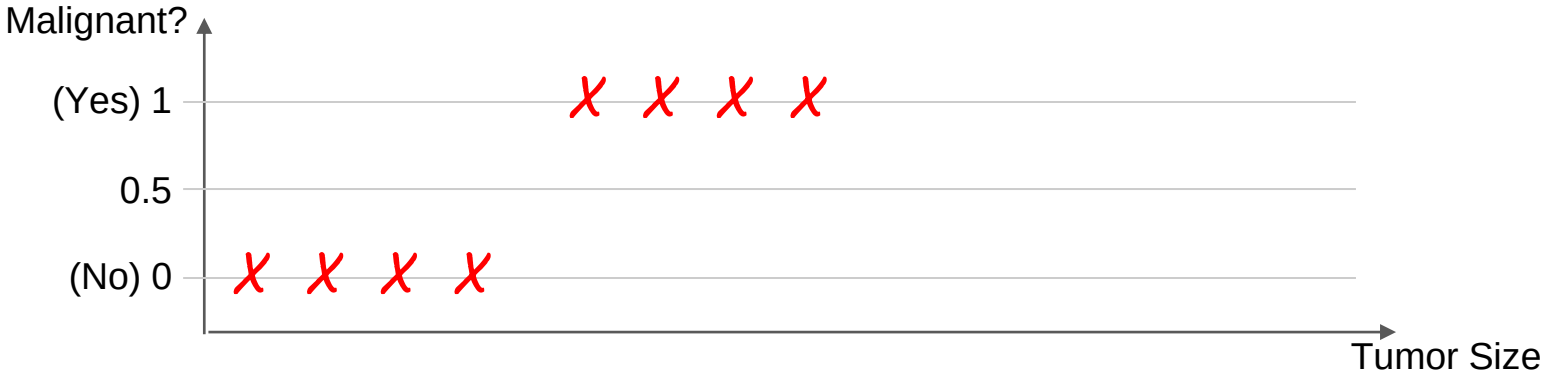
Linear Regression As A Classifier!

Why not use linear regression as a classifier?



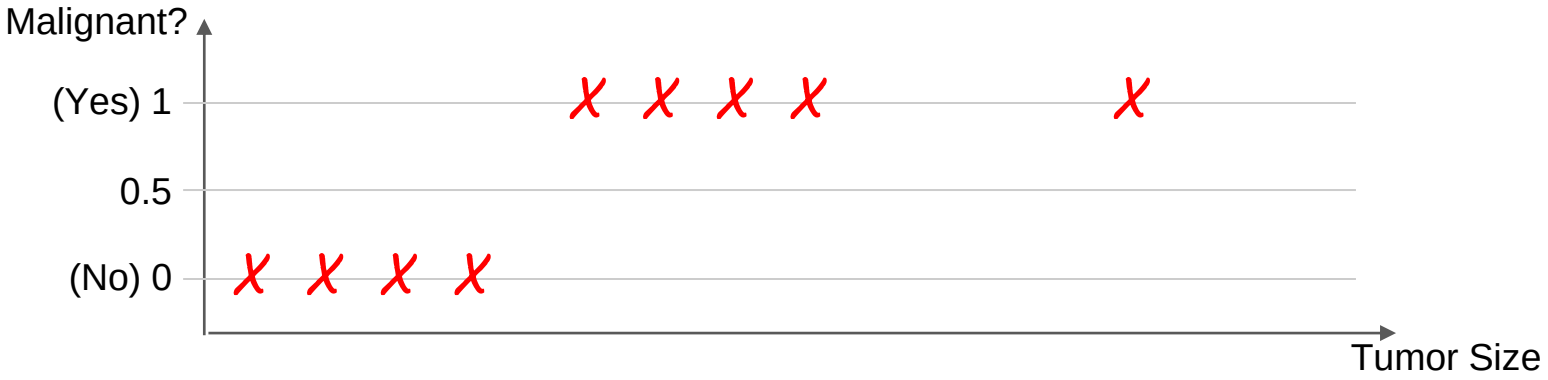
Linear Regression As A Classifier!

Why not use linear regression as a classifier?



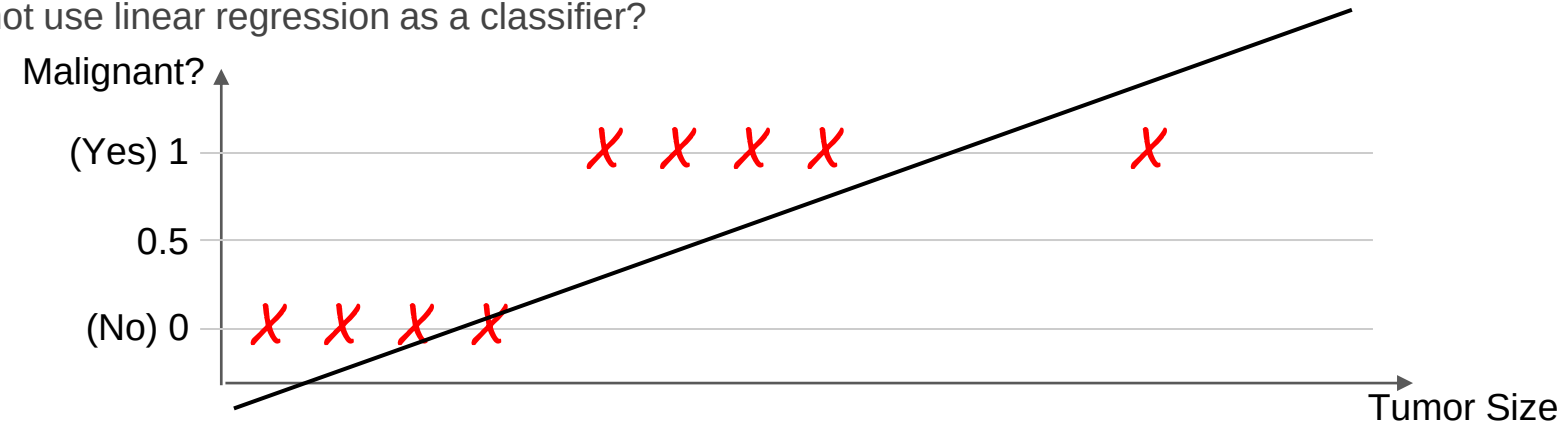
Linear Regression As A Classifier!

Why not use linear regression as a classifier?



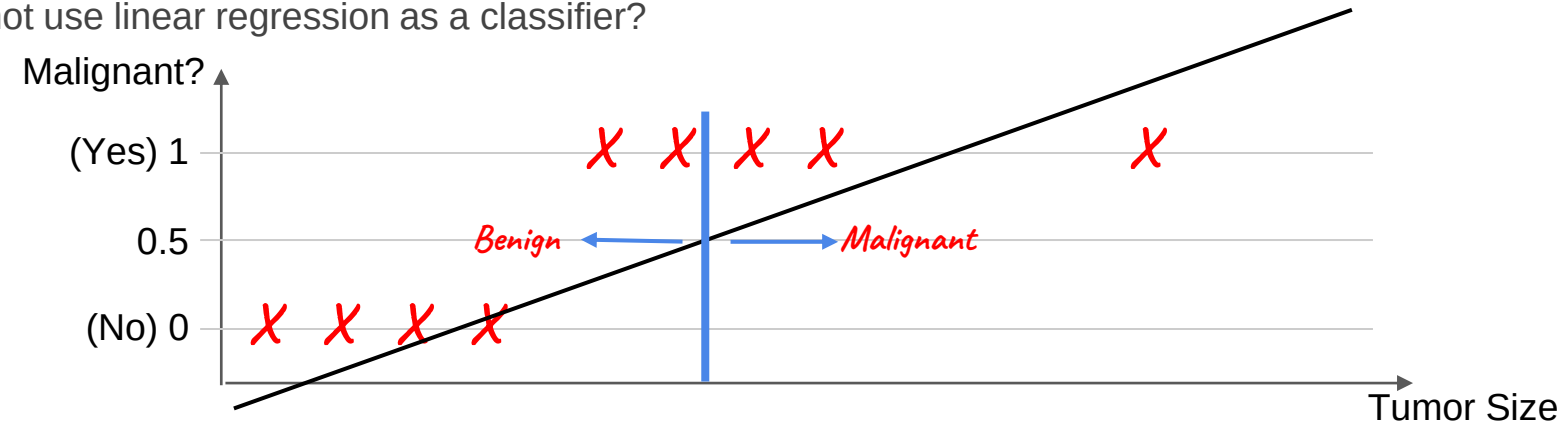
Linear Regression As A Classifier!

Why not use linear regression as a classifier?



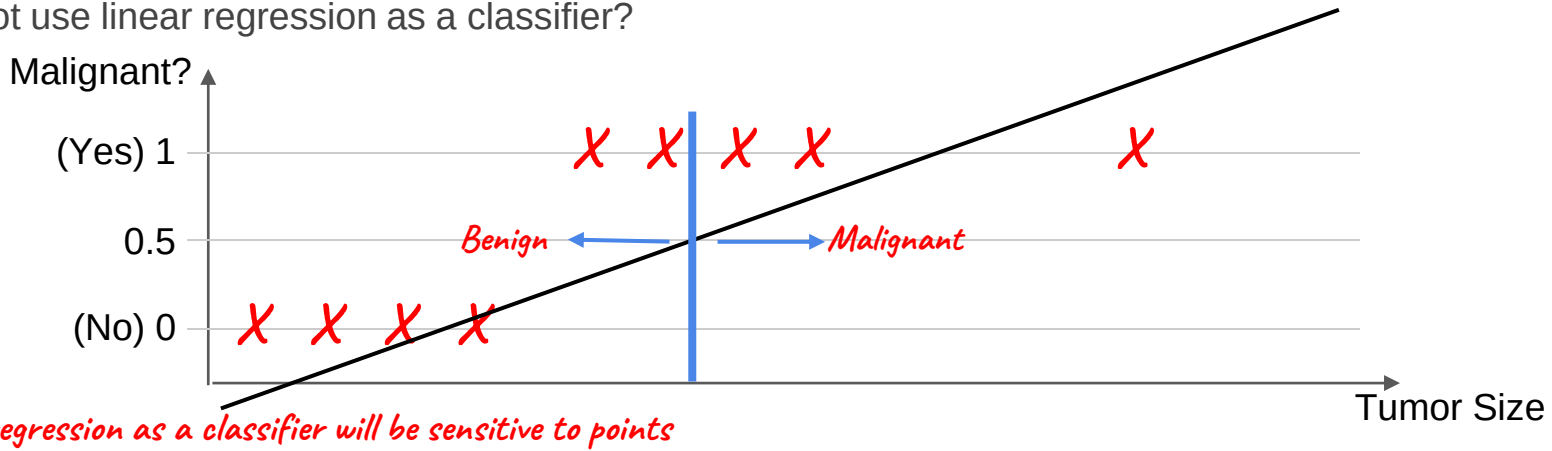
Linear Regression As A Classifier!

Why not use linear regression as a classifier?



Linear Regression As A Classifier!

Why not use linear regression as a classifier?

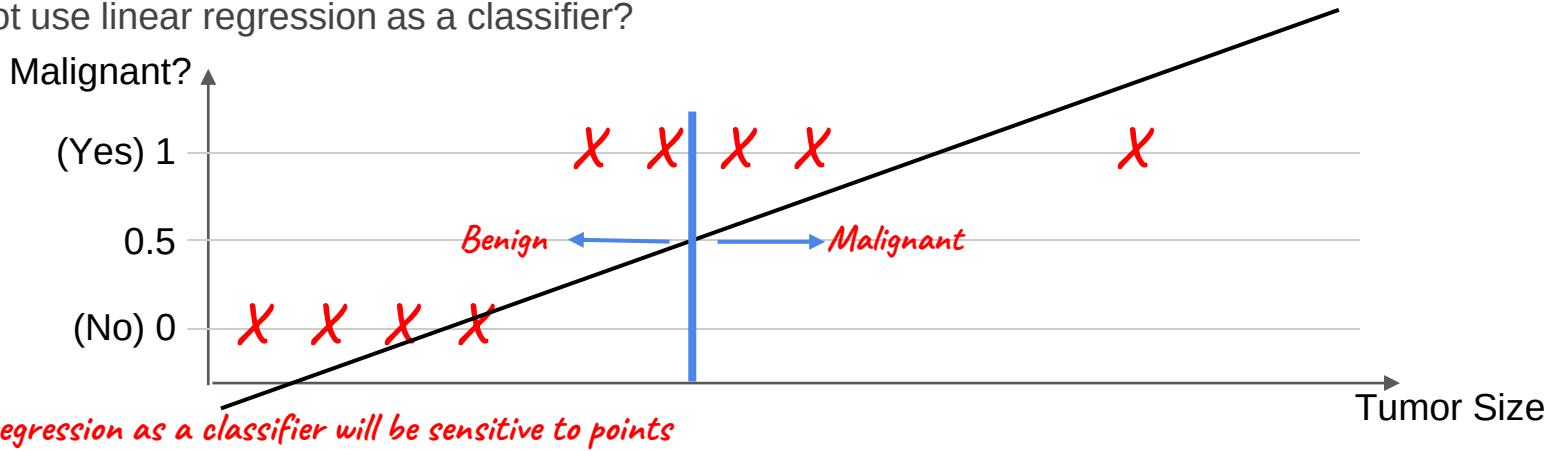


Linear regression as a classifier will be sensitive to points

As a classification problem:

Linear Regression As A Classifier!

Why not use linear regression as a classifier?

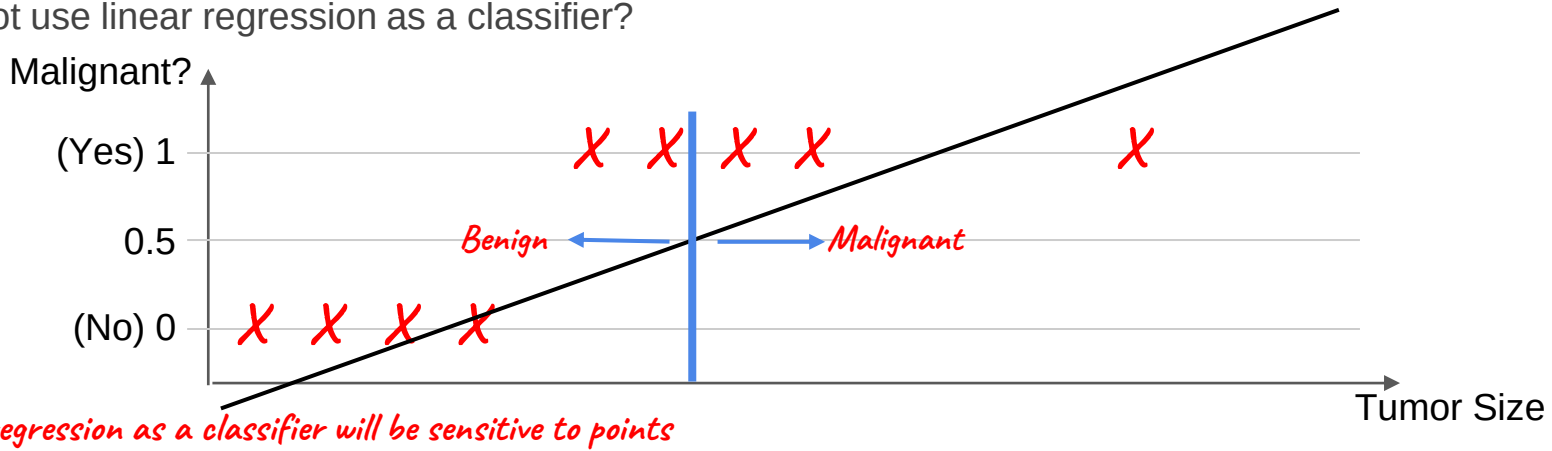


Linear regression as a classifier will be sensitive to points

As a classification problem:

Linear Regression As A Classifier!

Why not use linear regression as a classifier?

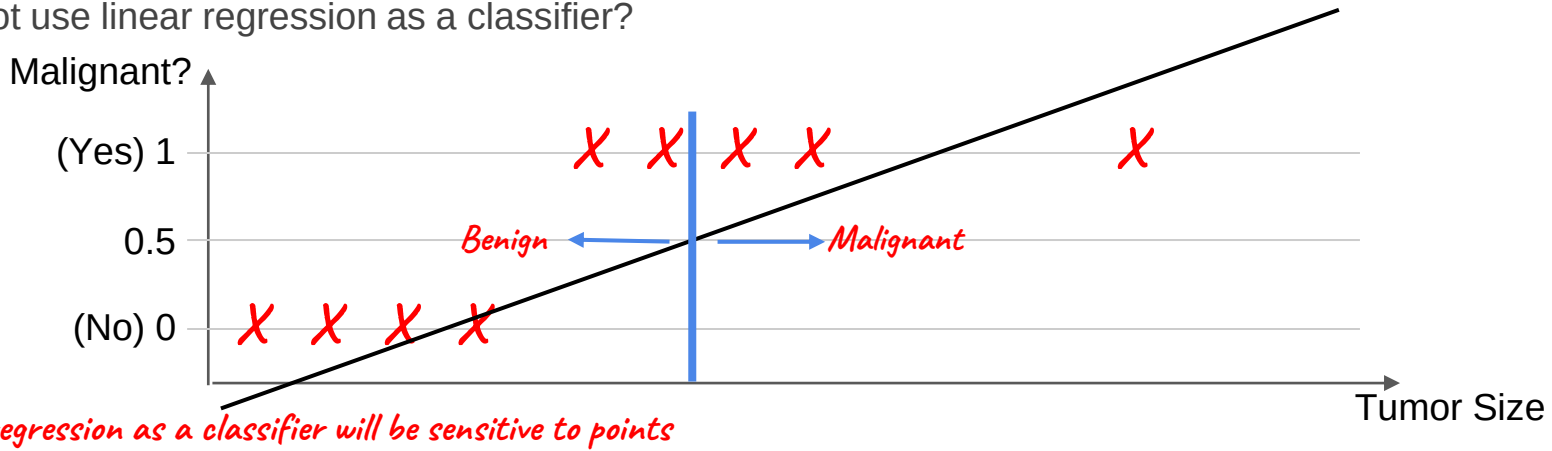


Linear regression as a classifier will be sensitive to points

As a classification problem:

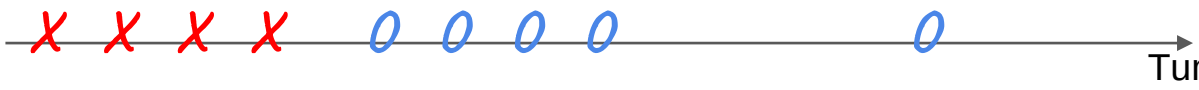
Linear Regression As A Classifier!

Why not use linear regression as a classifier?



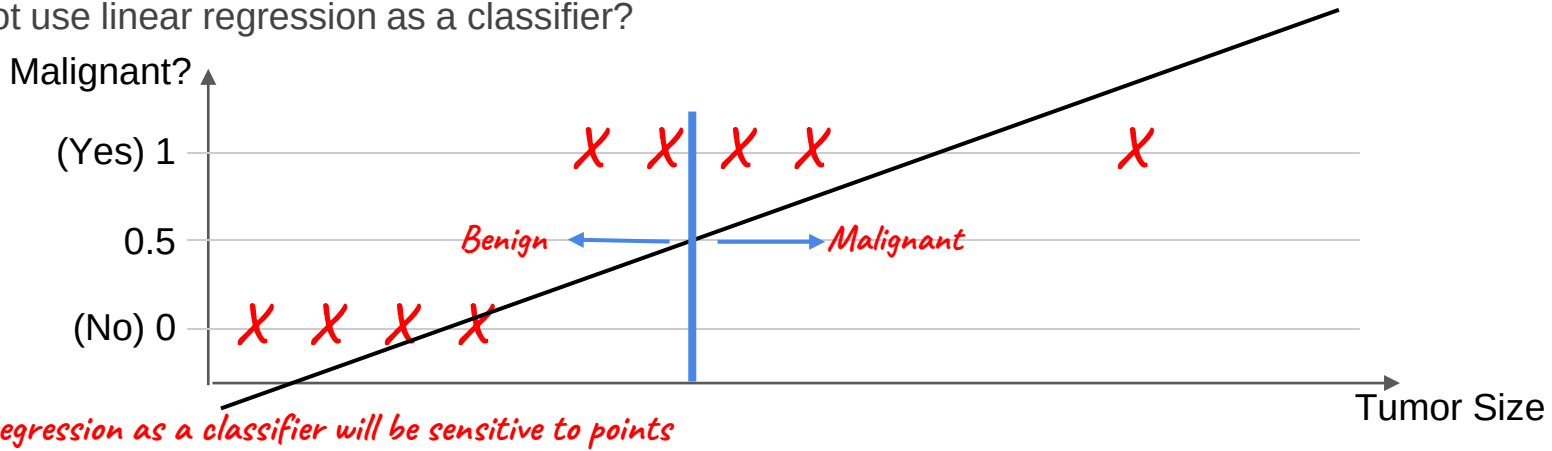
Linear regression as a classifier will be sensitive to points

As a classification problem:



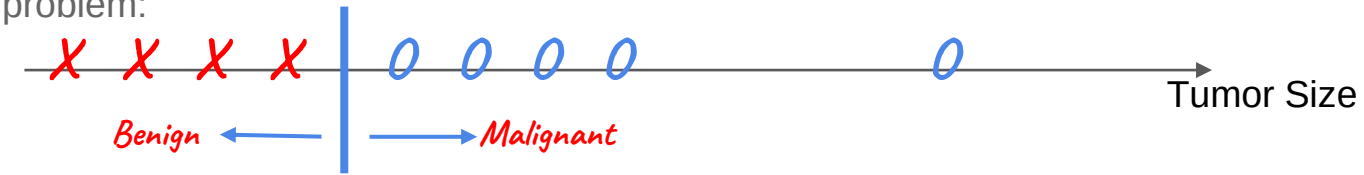
Linear Regression As A Classifier!

Why not use linear regression as a classifier?



Linear regression as a classifier will be sensitive to points

As a classification problem:



Lecture Overview

- **Throwback**
- **Linear Regression As A Classifier!**
- **Problem Definition**
- **Perceptron Classifier [Formula]**
- **Perceptron Classifier [error function]**
- **Perceptron Classifier [Training]**
- **Perceptron Classifier [SKlearn]**
- **Logistic regression vs Perceptron Classifier**
- **Logistic regression [cross entropy error]**
- **Logistic regression [Training]**
- **Logistic regression [SKlearn]**
- **Logistic regression [Multi-class Model]**

Star Wars!

Data	Prediction
 Aack aack aack!	Is this alien happy or sad?
 Beep beep!	
 Aack beep aack!	aack beep aack aack!
 Aack beep beep beep!	

Star Wars!

Imagine you invade a new planet

- The aliens have 2 words
  - Aack
  - Beep
- You want to know if alien is happy or sad
- You got this data form your experience with some aliens
- What do you notice about this data?
- How do we solve this?

Data		Prediction
	Aack aack aack!	Is this alien happy or sad?
	Beep beep!	
	Aack beep aack!	aack beep aack aack!
	Aack beep beep beep!	

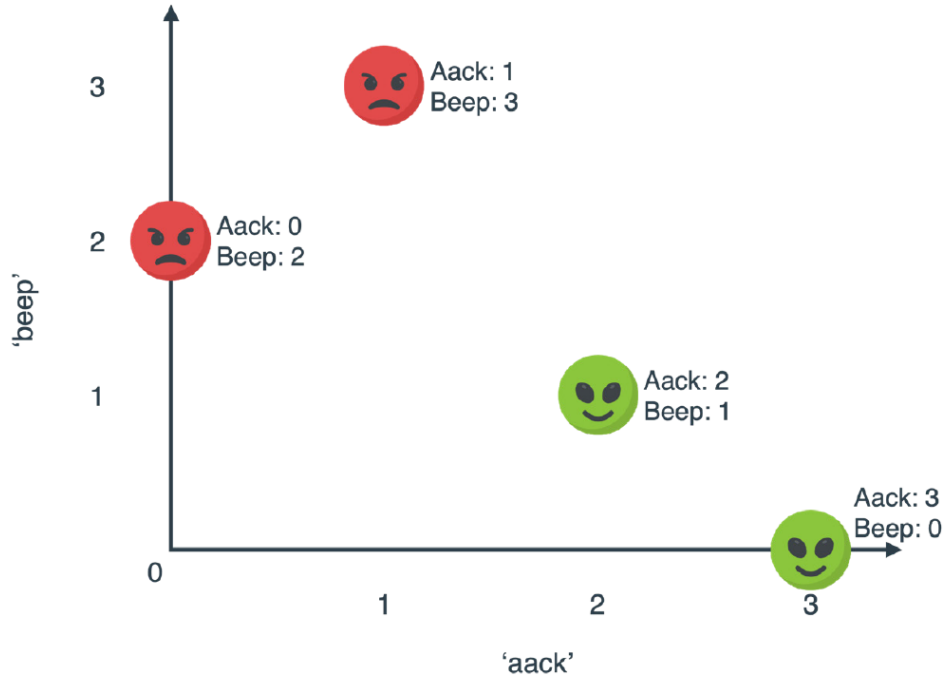
Star Wars!

Let’s build our dataset

Sentence	Aack	Beep	Mood
Aack aack aack!	3	0	Happy
Beep beep!	0	2	Sad
Aack beep aack!	2	1	Happy
Aack beep beep beep!	1	3	Sad

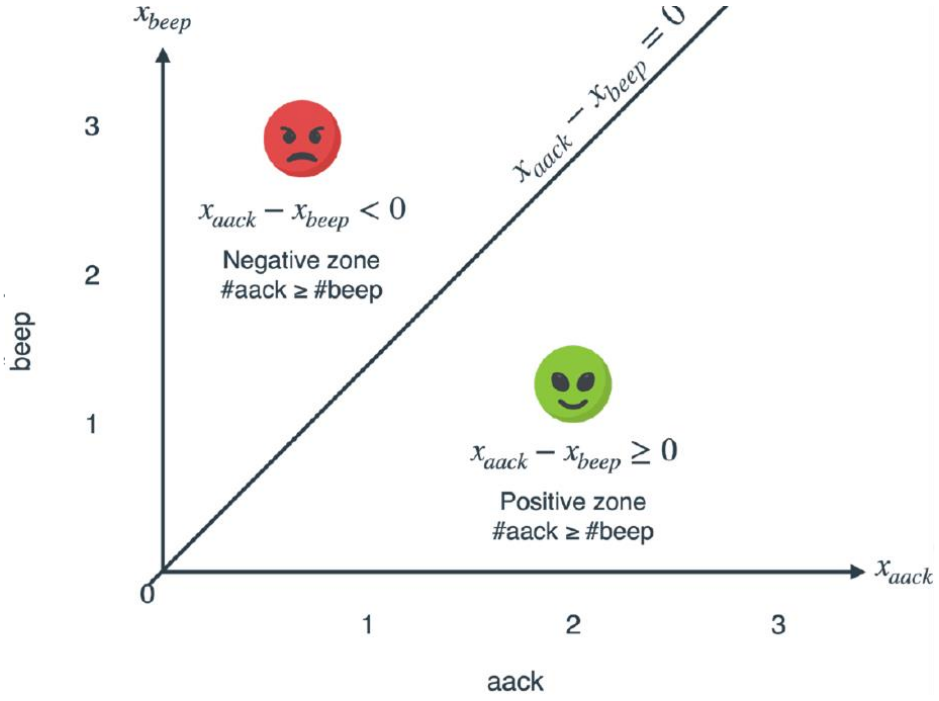
Star Wars!

Draw a line that classifies the data correctly.



Star Wars!

Draw a line that classifies the data correctly.



Star Wars!

Draw a line that classifies the data correctly.

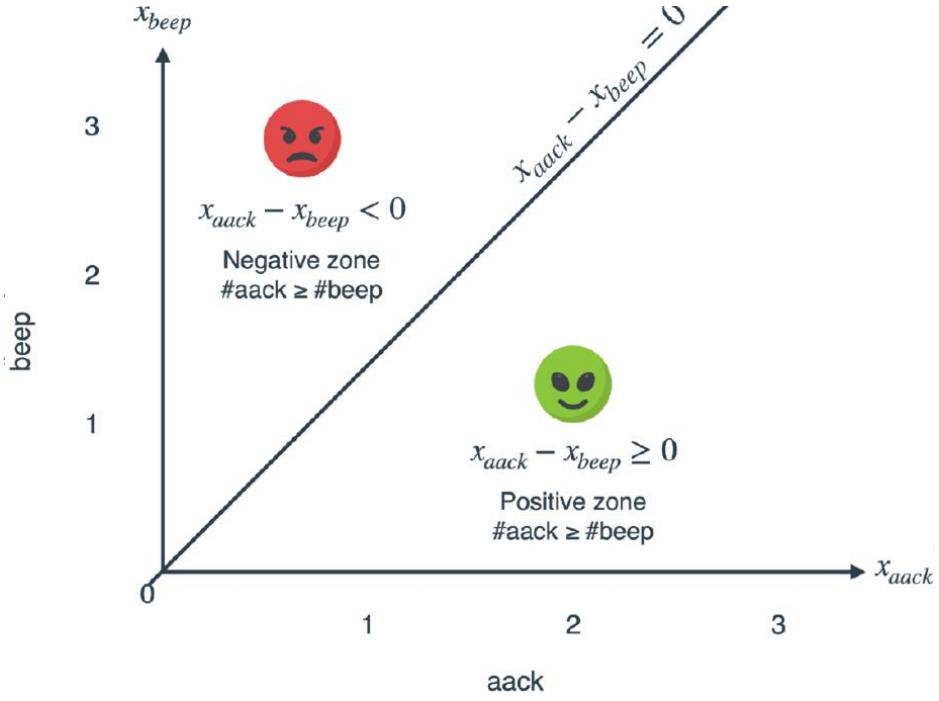
Line:

$X_{aack} = X_{beeb}$

Or

Line:

$X_{aack} - X_{beeb} = 0$



Another planet!

What do you notice about this planet?

Another planet!

What do you notice about this planet?

Another planet!

What do you notice about this planet?

$X_{crack} - X_{doink} - 3.5 = 0$

Happy:

$X_{crack} - X_{doink} - 3.5 >= 0$

Sad:

$X_{crack} - X_{doink} - 3.5 < 0$

Another planet!

What do you notice about this planet?

$X_{crack} - X_{doink} - 3.5 = 0$

Happy:

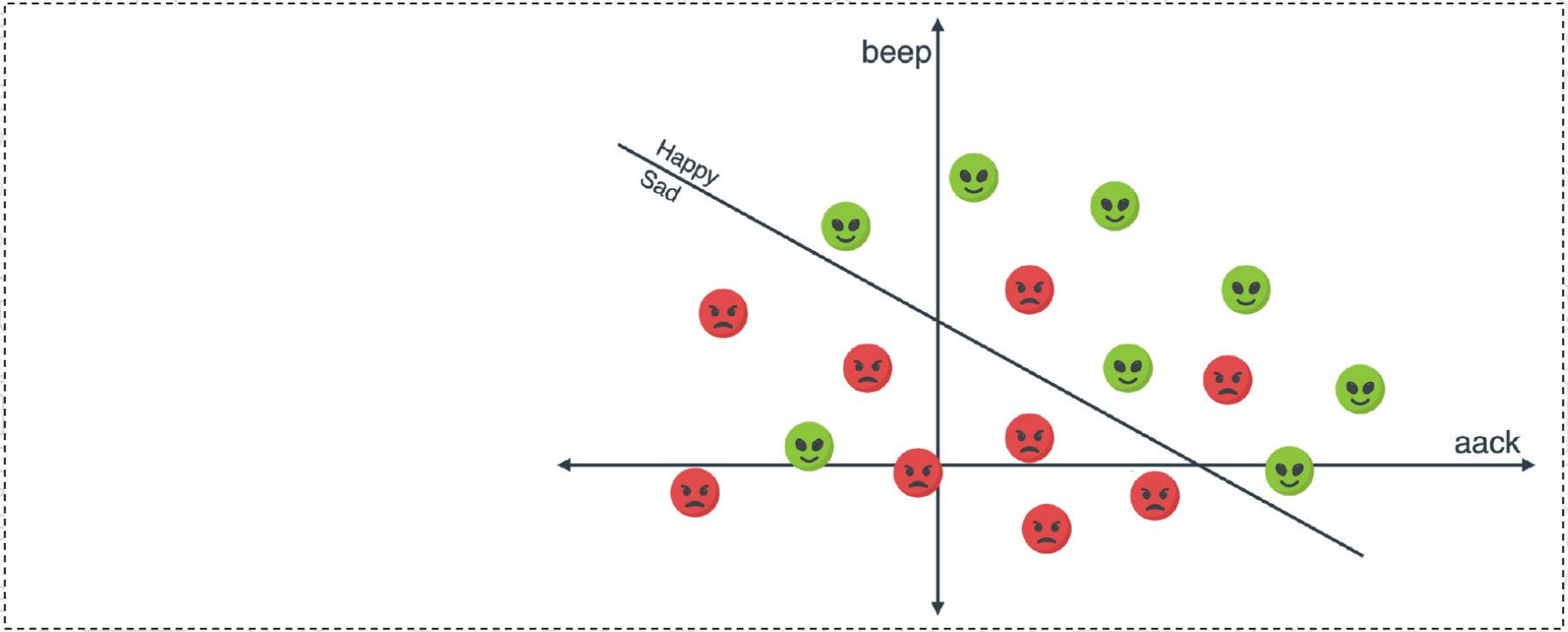
$X_{crack} - X_{doink} - 3.5 >= 0$

Sad:

$X_{crack} - X_{doink} - 3.5 < 0$

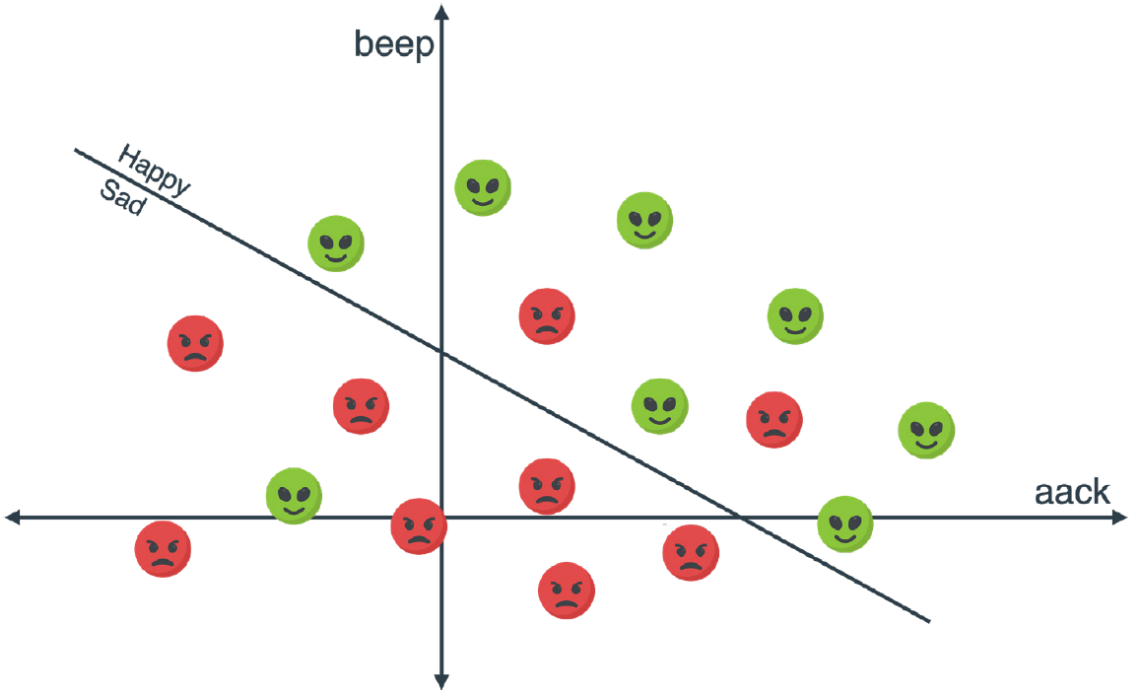
Is it always that simple?

Another dataset example



Another dataset example

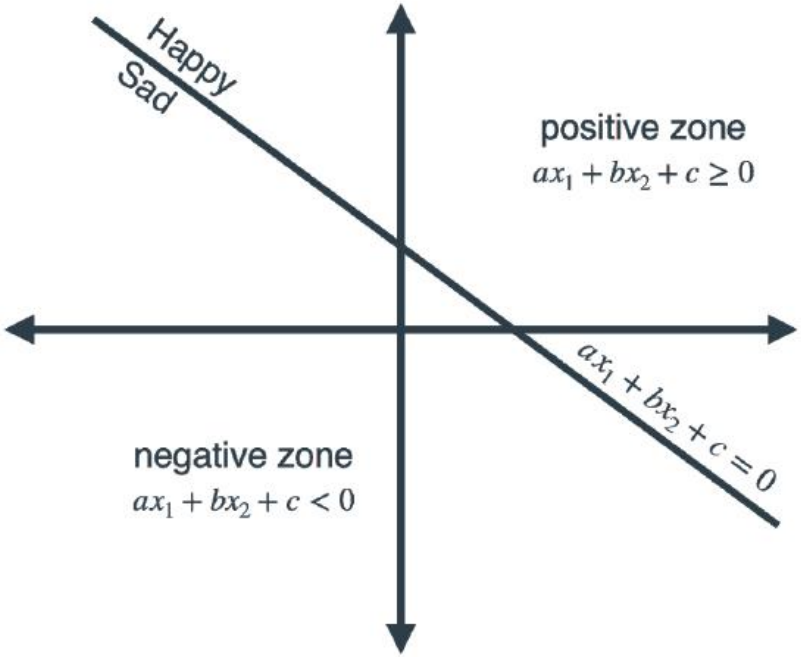
- Model could mistake some samples
- We try to find the most general model with least error



Lecture Overview

- Throwback
- Linear Regression As A Classifier!
- Problem Definition
- Perceptron Classifier [Formula]
- Perceptron Classifier [error function]
- Perceptron Classifier [Training]
- Perceptron Classifier [SKlearn]
- Logistic regression vs Perceptron Classifier
- Logistic regression [cross entropy error]
- Logistic regression [Training]
- Logistic regression [SKlearn]
- Logistic regression [Multi-class Model]

Perceptron Classifier [**Formula**]



Perceptron Classifier [**Formula**]

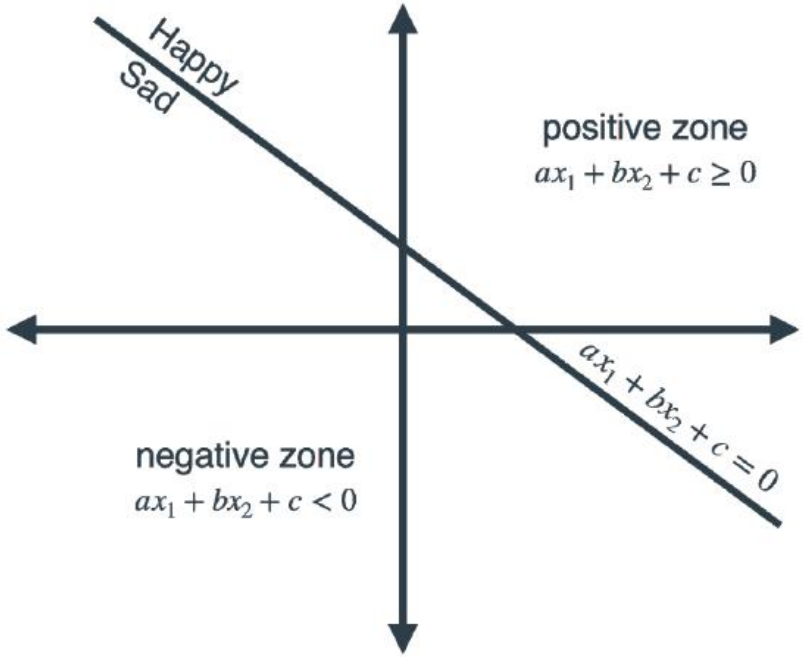
The general form of classifiers using line:
$ax_1 + bx_2 + c = 0$

This equation is different from the equation used in linear regression but they are equal.

What do a, b, and c represent?

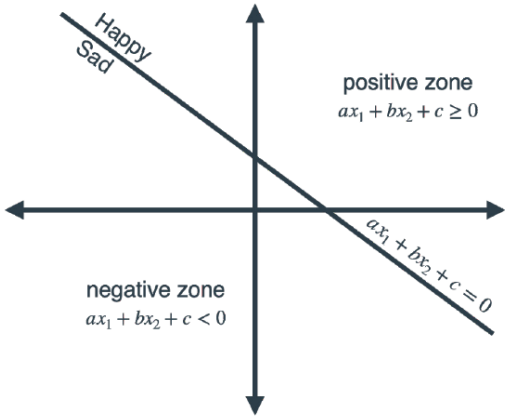
This formula could be extended if features more than 2.
Ex (3 features, plane model):
$ax_1 + bx_2 + cx_3 + d = 0$

So why we use this formula?



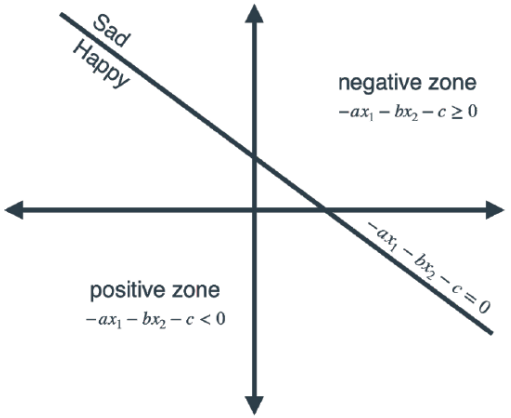
Perceptron Classifier [**Formula**]

Here is the answer...



Classifier with equation

$$ax_1 + bx_2 + c = 0$$

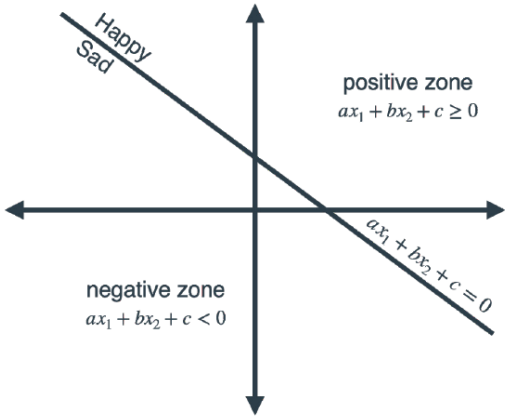


Classifier with equation

$$-ax_1 - bx_2 - c = 0$$

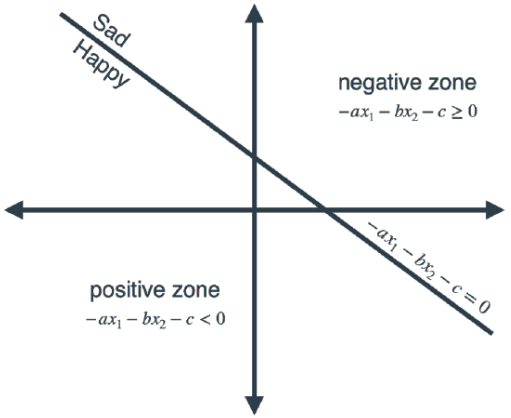
Perceptron Classifier [**Formula**]

Here is the answer...



Classifier with equation

$$ax_1 + bx_2 + c = 0$$

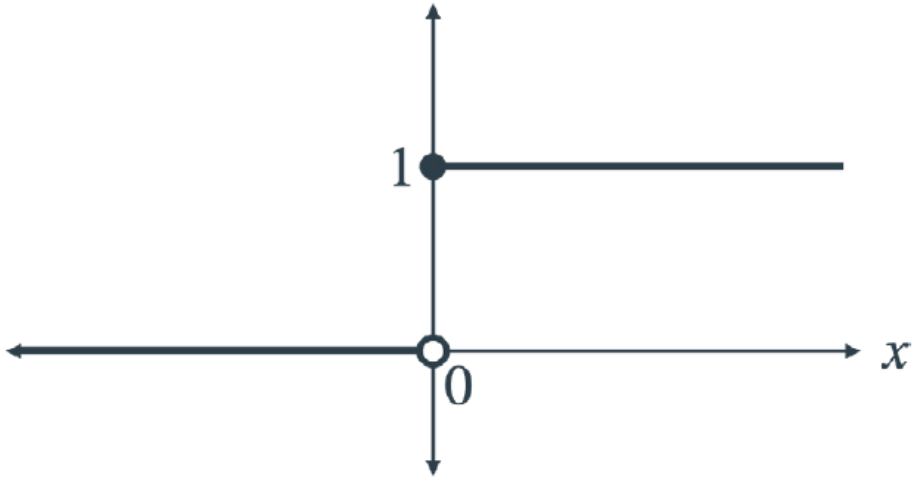


Classifier with equation

$$-ax_1 - bx_2 - c = 0$$

How to take the decision?

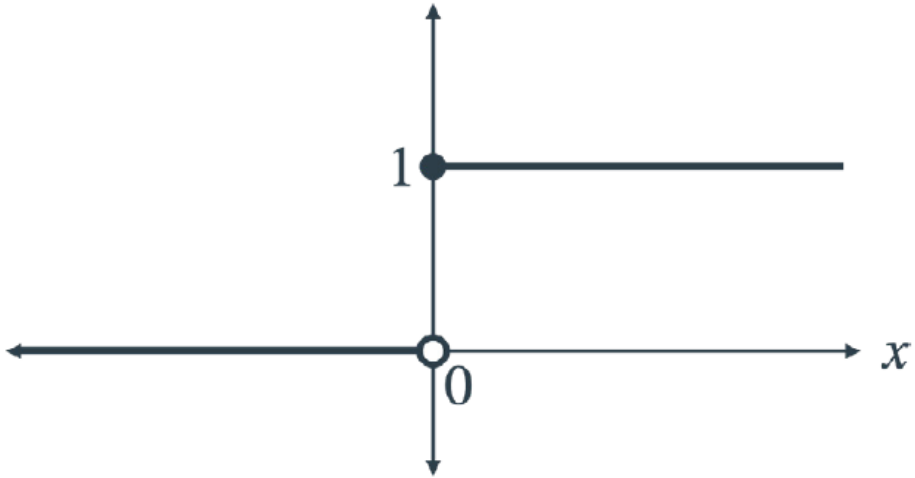
Perceptron Classifier [**Step function**]



Perceptron Classifier [**Step function**]

step(x) = 1 if x>=0
step(x) =0 if x<0

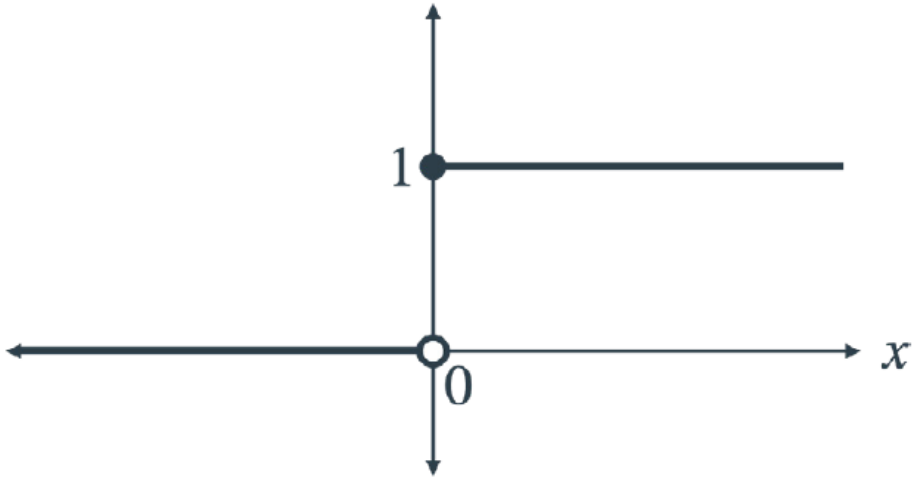
Classifier output:
y'= step($ax_1 + bx_2 + c$)
1: happy
0: sad



Perceptron Classifier [**Step function**]

step(x) = 1 if x>=0
step(x) =0 if x<0

Classifier output:
y'= step($ax_1 + bx_2 + c$)
1: happy
0: sad



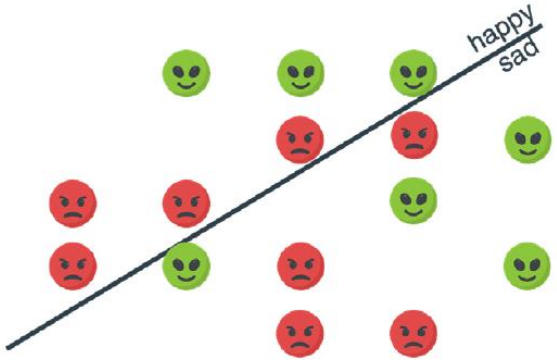
How to compare between 2 models?

Lecture Overview

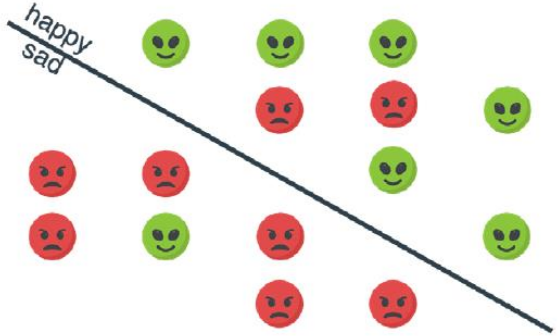
- Throwback
- Linear Regression As A Classifier!
- Problem Definition
- Perceptron Classifier [Formula]
- Perceptron Classifier [error function]
- Perceptron Classifier [Training]
- Perceptron Classifier [SKlearn]
- Logistic regression vs Perceptron Classifier
- Logistic regression [cross entropy error]
- Logistic regression [Training]
- Logistic regression [SKlearn]
- Logistic regression [Multi-class Model]

Perceptron Classifier [**Compare Models (error function)**]

1- Number of errors



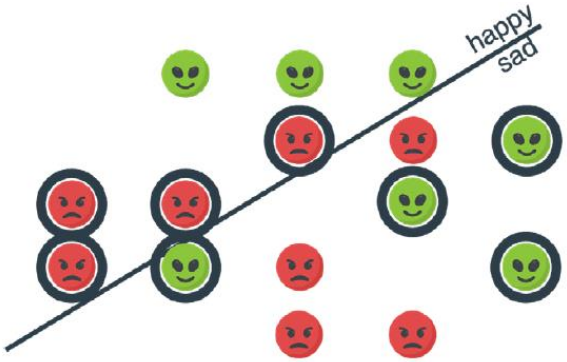
Bad classifier



Good classifier

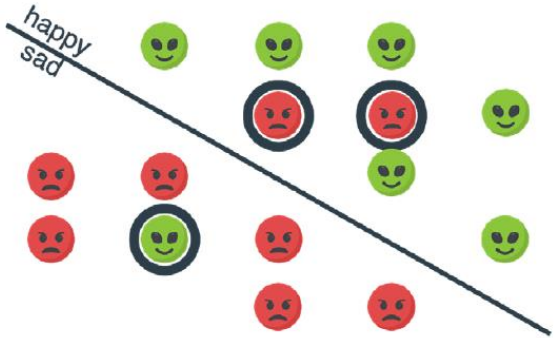
Perceptron Classifier [**Compare Models (error function)**]

1- Number of errors



Bad classifier

Error: 8



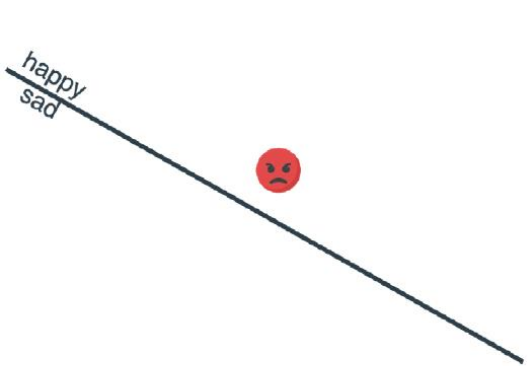
Good classifier

Error: 3

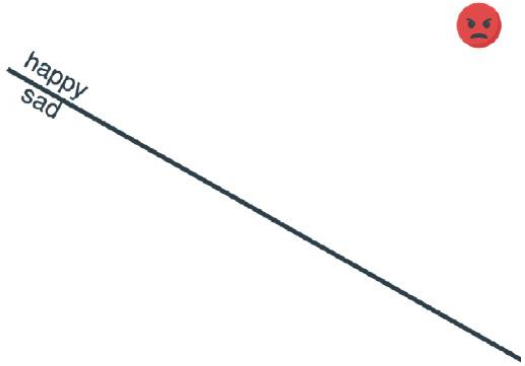
Perceptron Classifier [**Compare Models (error function)**]

1- Number of errors

Problem with this method:
- Doesn’t have a degree of true and false => don’t have a measure of converging



Poorly misclassified

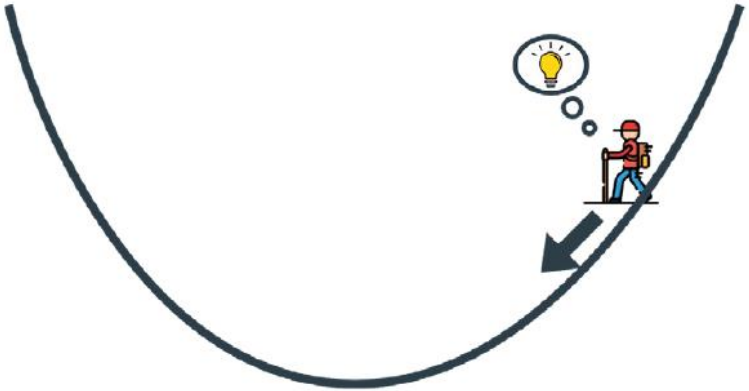


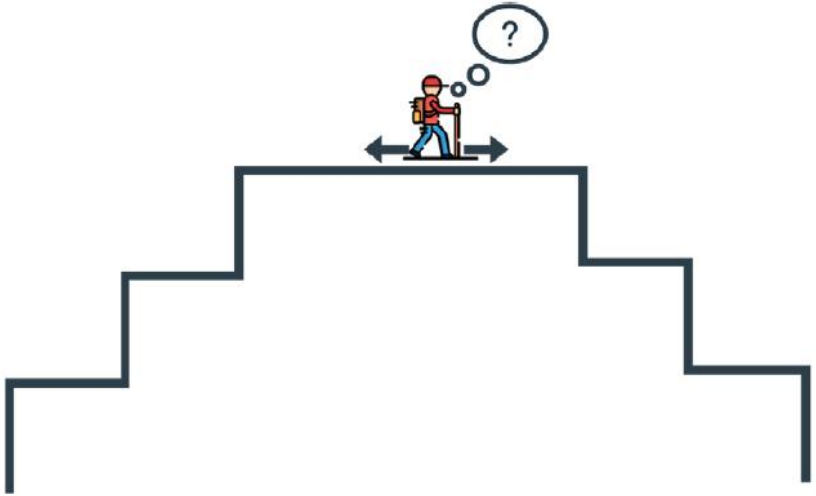
Very poorly misclassified

Perceptron Classifier [**Compare Models (error function)**]

1- Number of errors

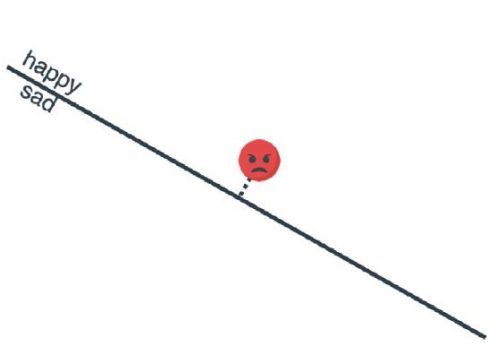
Problem with this method:
- Doesn’t have a degree of true and false => don’t have a measure of converging



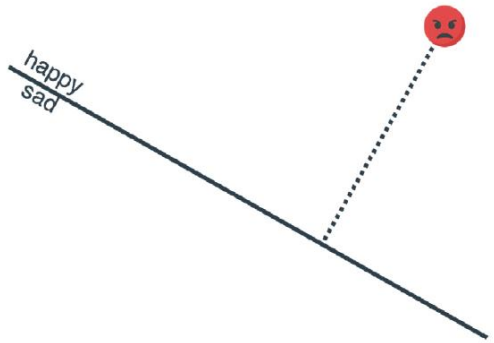


Perceptron Classifier [**Compare Models (error function)**]

2- Distance



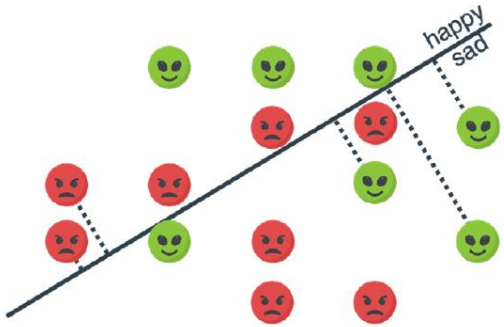
Small distance
Small error



Large distance
Large error

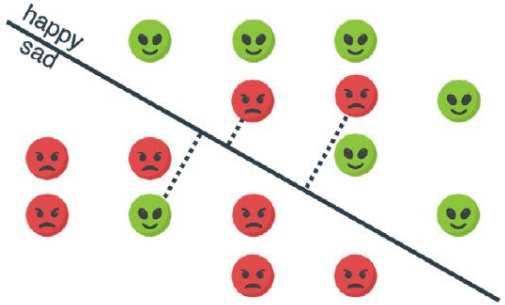
Perceptron Classifier [**Compare Models (error function)**]

2- Distance



Bad classifier

Error: .....



Good classifier

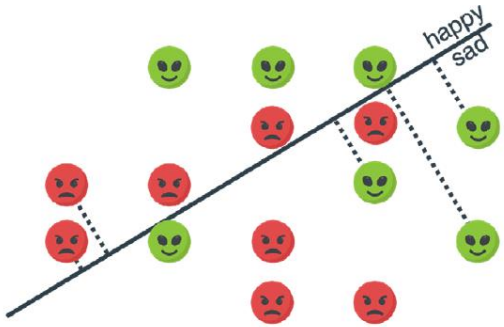
Error: .....

Perceptron Classifier [**Compare Models (error function)**]

2- Distance

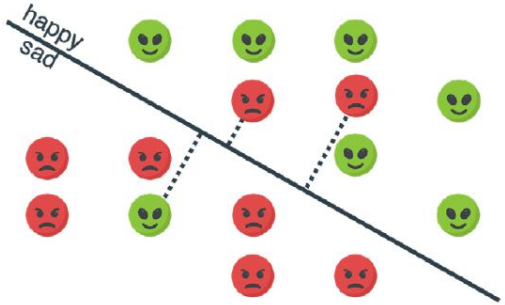
Problem with this method:

- Mathematically complicated



Bad classifier

Error: .....



Good classifier

Error: .....

Perceptron Classifier [**Compare Models (error function)**]

Perceptron Classifier [**Compare Models (error function)**]

3- Score

Properties:

- The points in the boundary have a score of 0.
- The points in the positive zone have positive scores.
- The points in the negative zone have negative scores.
- The points close to the boundary have scores of low magnitude.
- The points far from the boundary have scores of high magnitude.

Perceptron Classifier [**Compare Models (error function)**]

3- Score

Properties:

- The points in the boundary have a score of 0.
- The points in the positive zone have positive scores.
- The points in the negative zone have negative scores.
- The points close to the boundary have scores of low magnitude.
- The points far from the boundary have scores of high magnitude.

$$ax_1 + bx_2 + c$$

Perceptron Classifier [**Compare Models (error function)**]

3- Score

Properties:

- The points in the boundary have a score of 0.
- The points in the positive zone have positive scores.
- The points in the negative zone have negative scores.
- The points close to the boundary have scores of low magnitude.
- The points far from the boundary have scores of high magnitude.

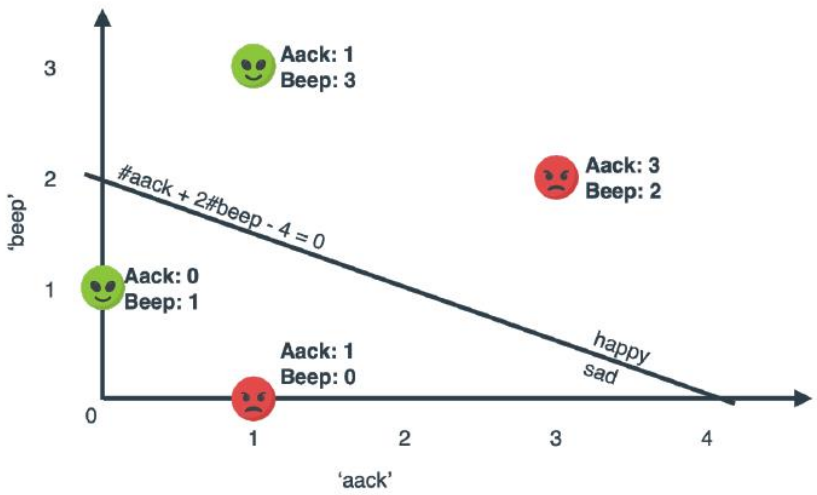
Score error:

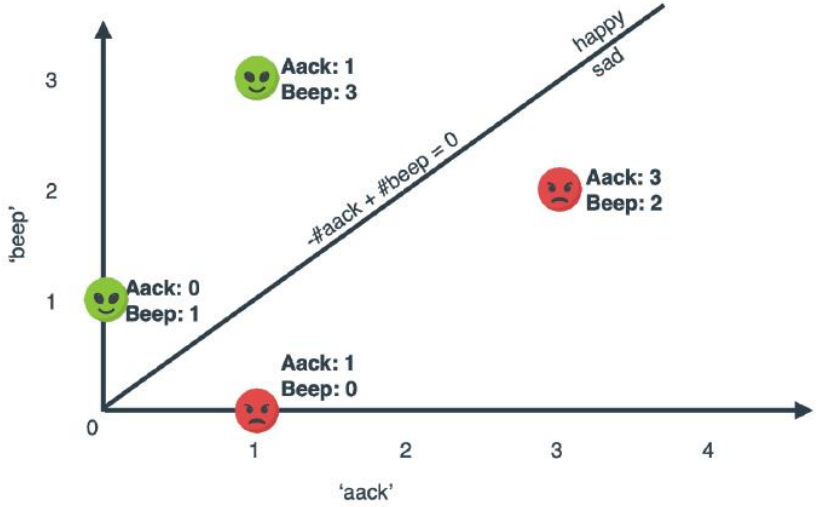
• If the sentence is correctly classified, the error is 0.

• If the sentence is misclassified, the error is $|ax_1 + bx_2 + c|$.

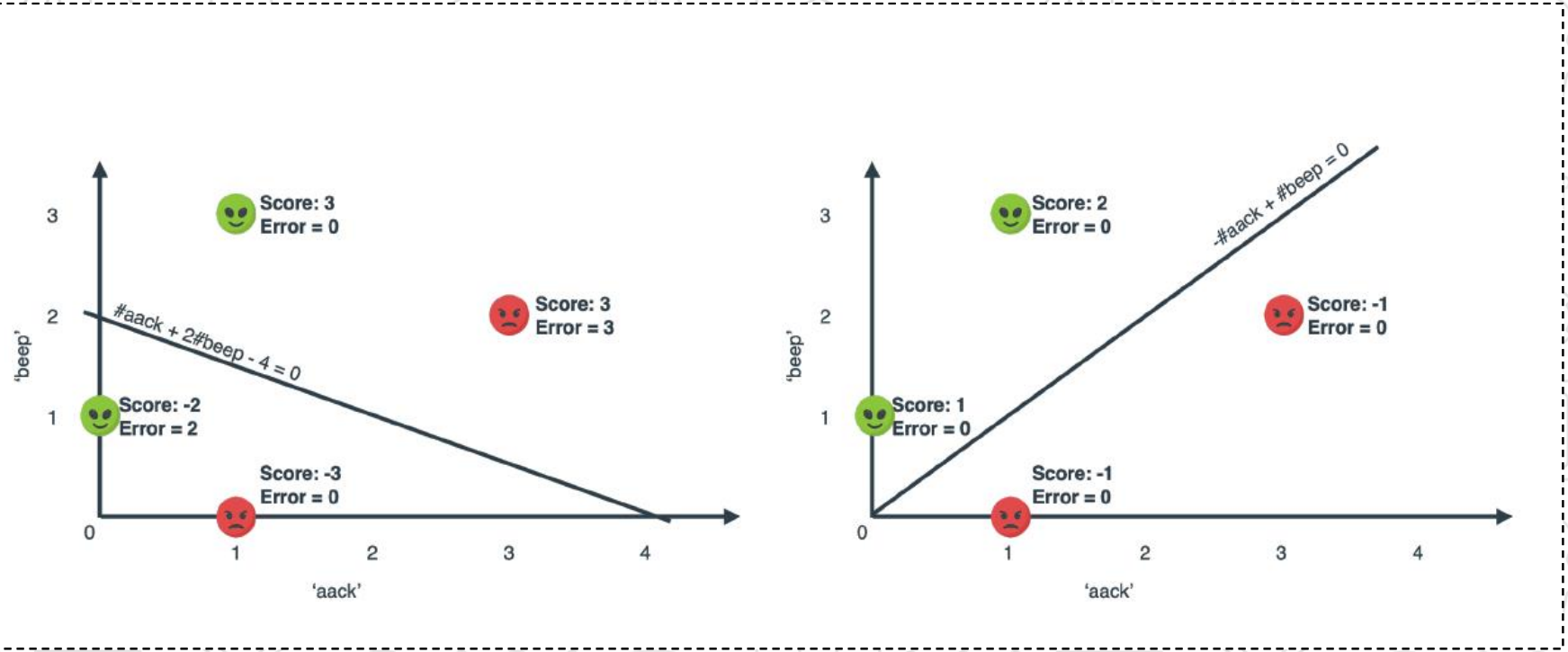
$$ax_1 + bx_2 + c$$

Perceptron Classifier [**Score error function example**]





Perceptron Classifier [**Score error function example**]



Perceptron Classifier [Score error function example]

Sentence (x_{aack}, x_{beep})	Label y	Classifier 1 score $1x_{aack}+2x_{beep}-4$	Classifier 1 prediction step($1x_{aack}+2x_{beep}-4$)	Classifier 1 error	Classifier 2 score $-x_{aack}+x_{beep}$	Classifier 2 prediction step($-x_{aack}+x_{beep}$)	Classifier 2 error
(1,0)	Sad (0)	-3	0 (correct)	0	-1	0 (correct)	0
(0,1)	Happy (1)	-2	0 (incorrect)	2	1	1 (correct)	0
(1,3)	Happy (1)	3	1 (correct)	3	2	1 (correct)	0
(3,2)	Sad (0)	3	1 (incorrect)	0	-1	0 (correct)	0
Mean				1.25			0

Prediction and Error Function

Use colab to open this github notebook:

“s7s/machine_learning_1/perceptron_algorithm/Coding_perceptron_algorithm.ipynb”

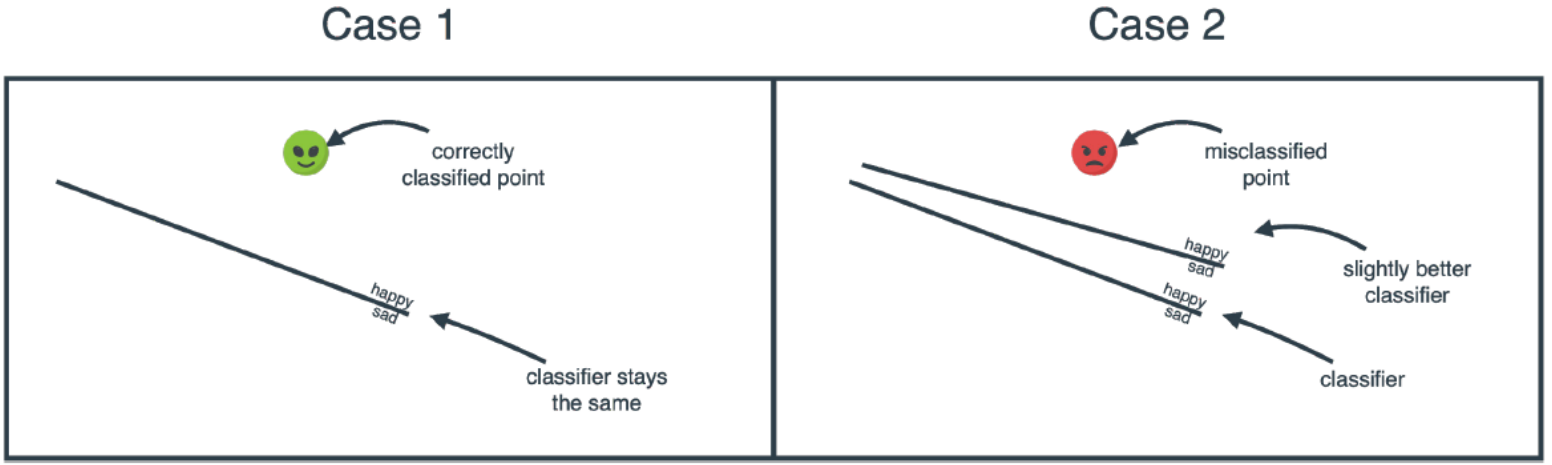
Lecture Overview

- Throwback
- Linear Regression As A Classifier!
- Problem Definition
- Perceptron Classifier [Formula]
- Perceptron Classifier [error function]
- Perceptron Classifier [Training]
- Perceptron Classifier [SKlearn]
- Logistic regression vs Perceptron Classifier
- Logistic regression [cross entropy error]
- Logistic regression [Training]
- Logistic regression [SKlearn]
- Logistic regression [Multi-class Model]

Perceptron Classifier [Training]

- Case 1: If the point is correctly classified, leave the line as it is.

- Case 2: If the point is incorrectly classified, that means it produces a positive error. Adjust the weights and the bias a small amount so that this error slightly decreases.



Perceptron Classifier [Training]

Procedure:

- The prediction the perceptron makes at the point is

$$\hat{y} = step(ax_1 + bx_2 + c).$$

Perceptron Classifier [Training]

Procedure:

- The prediction the perceptron makes at the point is

$$\hat{y} = step(ax_1 + bx_2 + c).$$

- **Case 1:** If $\hat{y} = y$:
  - **Return** the input perceptron

Perceptron Classifier [Training]

Procedure:

- The prediction the perceptron makes at the point is

$$\hat{y} = step(ax_1 + bx_2 + c).$$

- **Case 1:** If $\hat{y} = y$:
  - **Return** the input perceptron
- **Case 2:** If $\hat{y} = 1$ and $y = 0$:
  - **Return** the perceptron with the following weights and bias:
    - $a' = a - \eta$
    - $b' = b - \eta$
    - $c' = c - \eta.$

Perceptron Classifier [Training]

Procedure:

- The prediction the perceptron makes at the point is

$$\hat{y} = step(ax_1 + bx_2 + c).$$

- **Case 1:** If $\hat{y} = y$:
  - **Return** the input perceptron
- **Case 2:** If $\hat{y} = 1$ and $y = 0$:
  - **Return** the perceptron with the following weights and bias:
    - $a' = a - \eta x_1$
    - $b' = b - \eta x_2$
    - $c' = c - \eta.$

Perceptron Classifier [Training]

Procedure:

- The prediction the perceptron makes at the point is

$$\hat{y} = step(ax_1 + bx_2 + c).$$

- **Case 1:** If $\hat{y} = y$:
  - **Return** the input perceptron
- **Case 2:** If $\hat{y} = 1$ and $y = 0$:
  - **Return** the perceptron with the following weights and bias:
    - $a' = a - \eta x_1$
    - $b' = b - \eta x_2$
    - $c' = c - \eta.$
- **Case 3:** If $\hat{y} = 0$ and $y = 1$:
  - **Return** the perceptron with the following weights and bias:
    - $a' = a + \eta x_1$
    - $b' = b + \eta x_2$
    - $c' = c + \eta.$

Perceptron Classifier [Training]

Procedure:

- The prediction the perceptron makes at the point is

$$\hat{y} = step(ax_1 + bx_2 + c).$$

- **Case 1:** If $\hat{y} = y$:
  - **Return** the input perceptron
- **Case 2:** If $\hat{y} = 1$ and $y = 0$:
  - **Return** the perceptron with the following weights and bias:
    - $a' = a - \eta x_1$
    - $b' = b - \eta x_2$
    - $c' = c - \eta.$
- **Case 3:** If $\hat{y} = 0$ and $y = 1$:
  - **Return** the perceptron with the following weights and bias:
    - $a' = a + \eta x_1$
    - $b' = b + \eta x_2$
    - $c' = c + \eta.$

Combining all cases in one case:

- $a' = a + \eta(y - \hat{y})x_1$
- $b' = b + \eta(y - \hat{y})x_2$
- $c' = c + \eta(y - \hat{y}).$

Perceptron Classifier [Training]

Procedure:

- The prediction the perceptron makes at the point is

$$\hat{y} = step(ax_1 + bx_2 + c).$$

- **Case 1:** If $\hat{y} = y$:
  - **Return** the input perceptron
- **Case 2:** If $\hat{y} = 1$ and $y = 0$:
  - **Return** the perceptron with the following weights and bias:
    - $a' = a - \eta x_1$
    - $b' = b - \eta x_2$
    - $c' = c - \eta.$
- **Case 3:** If $\hat{y} = 0$ and $y = 1$:
  - **Return** the perceptron with the following weights and bias:
    - $a' = a + \eta x_1$
    - $b' = b + \eta x_2$
    - $c' = c + \eta.$

Perceptron Classifier [Training]

Procedure:

- The prediction the perceptron makes at the point is

$$\hat{y} = step(ax_1 + bx_2 + c).$$

- **Case 1:** If $\hat{y} = y$:
  - **Return** the input perceptron
- **Case 2:** If $\hat{y} = 1$ and $y = 0$:
  - **Return** the perceptron with the following weights and bias:
    - $a' = a - \eta x_1$
    - $b' = b - \eta x_2$
    - $c' = c - \eta.$
- **Case 3:** If $\hat{y} = 0$ and $y = 1$:
  - **Return** the perceptron with the following weights and bias:
    - $a' = a + \eta x_1$
    - $b' = b + \eta x_2$
    - $c' = c + \eta.$

Case 2,3 mathematically:

Error: $E = |ax_1 + bx_2 + c|$

a'=a - η (slope)

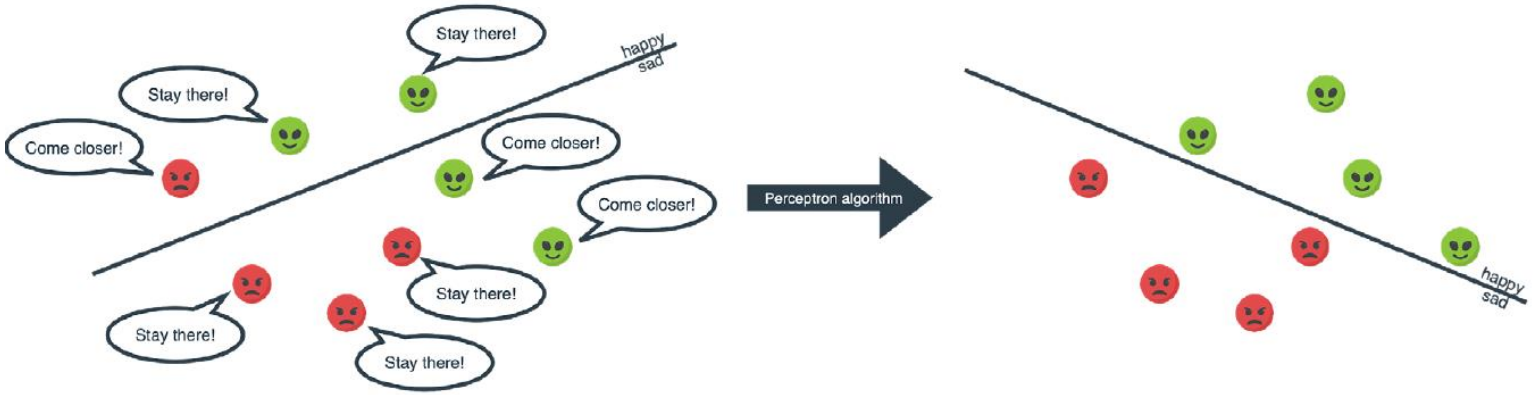
a'=a - η $\partial E/\partial a$

$\partial E/\partial a$ = sign($ax_1 + bx_2 + c$) * X_1

a'=a - η * sign($ax_1 + bx_2 + c$) * X_1

As well b and c..

Perceptron Classifier [**Training**]



Weights update using one point

Use colab to open this github notebook:

“s7s/machine_learning_1/perceptron_algorithm/Coding_perceptron_algorithm.ipynb”

Lecture Overview

- Throwback
- Linear Regression As A Classifier!
- Problem Definition
- Perceptron Classifier [Formula]
- Perceptron Classifier [error function]
- Perceptron Classifier [Training]
- Perceptron Classifier [SKlearn]
- Logistic regression vs Perceptron Classifier
- Logistic regression [cross entropy error]
- Logistic regression [Training]
- Logistic regression [SKlearn]
- Logistic regression [Multi-class Model]

SKlearn

Use colab to open this github notebook:

“s7s/machine_learning_1/perceptron_algorithm/Coding_perceptron_algorithm.ipynb”

SKlearn

```
>>> from sklearn.datasets import load_digits
>>> from sklearn.linear_model import Perceptron
>>> X, y = load_digits(return_X_y=True)
>>> clf = Perceptron(tol=1e-3, random_state=0)
>>> clf.fit(X, y)
Perceptron()
>>> clf.score(X, y)
0.939...
```

Lecture Overview

- Throwback
- Linear Regression As A Classifier!
- Problem Definition
- Perceptron Classifier [Formula]
- Perceptron Classifier [error function]
- Perceptron Classifier [Training]
- Perceptron Classifier [SKlearn]
- Logistic regression vs Perceptron Classifier
- Logistic regression [cross entropy error]
- Logistic regression [Training]
- Logistic regression [SKlearn]
- Logistic regression [Multi-class Model]

Perceptron Classifier [**Drawbacks**]

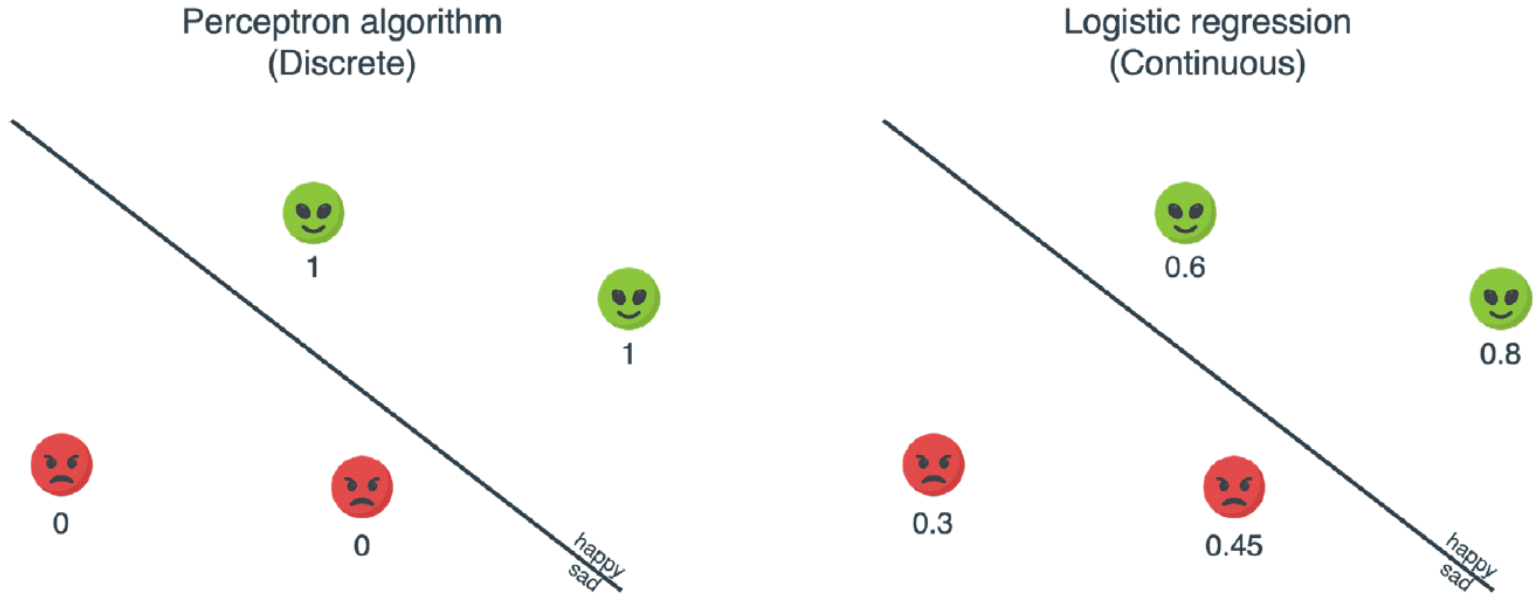
Perceptron Classifier [**Drawbacks**]

- The algorithm is designed for classifying two categories but not more.
- The step function is discrete which cause:
  - It is hard to have the derivative of it.
  - His answers offer very little information. For example, the sentence “I’m happy”, and the sentence “I’m thrilled, this is the best day of my life!”, are both happy sentences. However, the second one is much happier than the first one. A perceptron classifier will classify them both as ‘happy’.

So we need a continuous classifier and our **Logistic Regression Classifier** solve these problems.

Logistic regression vs Perceptron Classifier

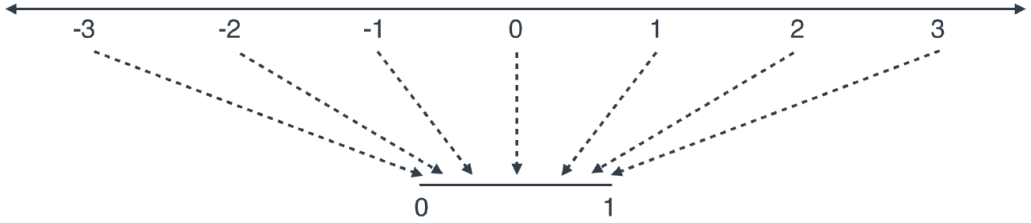
Logistic regression vs Perceptron Classifier



Why does it called logistic ‘regression’ while it is a classifier?!

Logistic regression [**Logistic function(Sigmoid)**]

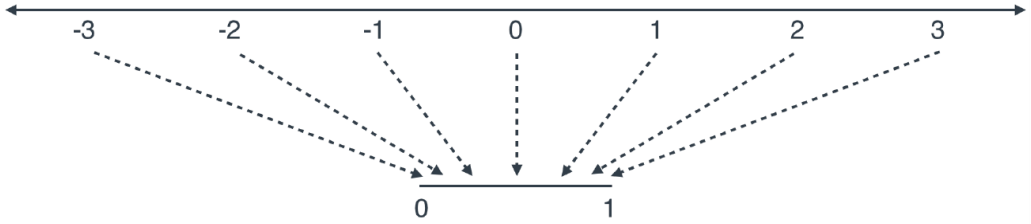
The output of this function should be continuous [0,1]



Logistic regression [**Logistic function(Sigmoid)**]

The output of this function should be continuous [0,1]

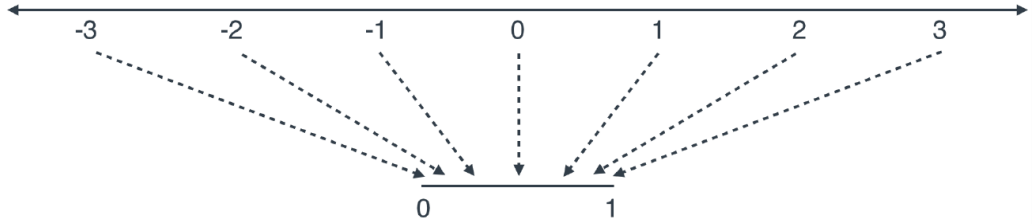
$$\sigma(x) = \frac{1}{1 + e^{-x}}$$



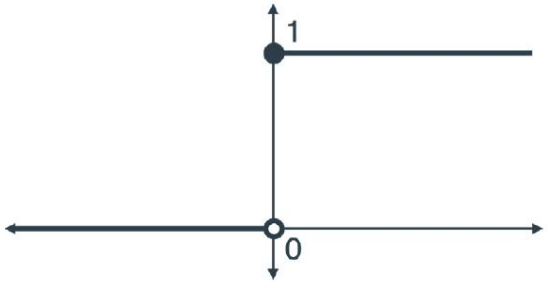
Logistic regression [**Logistic function(Sigmoid)**]

The output of this function should be continuous [0,1]

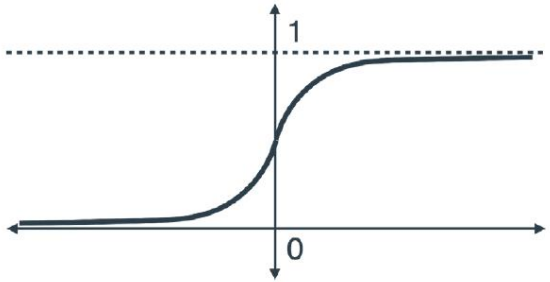
$$\sigma(x) = \frac{1}{1 + e^{-x}}$$



Step function (discrete)



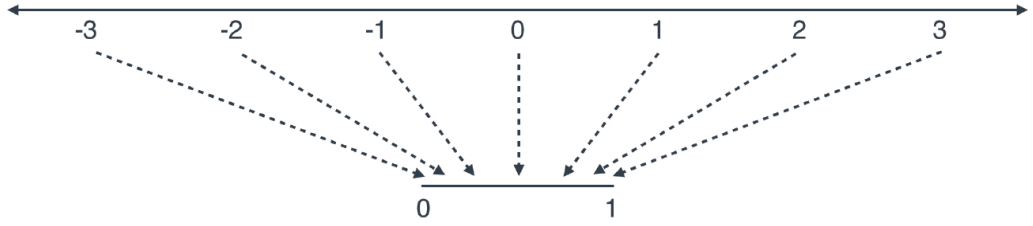
Sigmoid function (continuous)



Logistic regression [**Logistic function(Sigmoid)**]

The output of this function should be continuous [0,1]

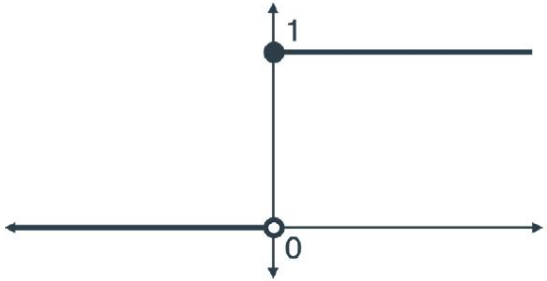
$$\sigma(x) = \frac{1}{1 + e^{-x}}$$



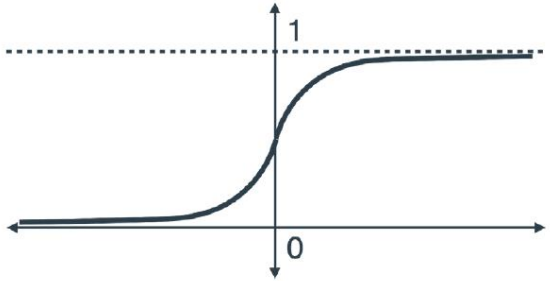
Another advantage of sigmoid function is the simple derivative:

Derivative of sigmoid(x)= sigmoid(x) * (1- sigmoid(x))

Step function (discrete)



Sigmoid function (continuous)



Logistic regression [**Star War Example**]

Prediction: $\hat{y} = \sigma(1 \cdot x_{aack} + 2 \cdot x_{beep} - 4)$

Logistic regression [**Star War Example**]

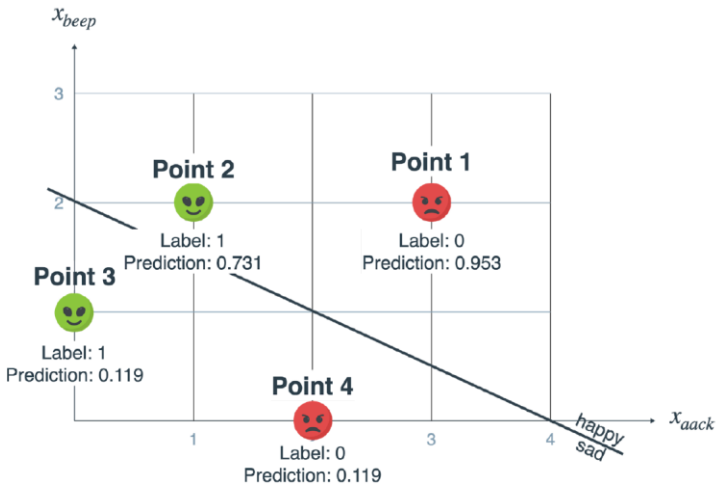
Prediction: $\hat{y} = \sigma(1 \cdot x_{aack} + 2 \cdot x_{beep} - 4)$

Sentence 1: $\hat{y} = \sigma(3 + 2 \cdot 2 - 4) = \sigma(3) = 0.953.$

Sentence 2: $\hat{y} = \sigma(1 + 2 \cdot 2 - 4) = \sigma(1) = 0.731.$

Sentence 3: $\hat{y} = \sigma(0 + 2 \cdot 1 - 4) = \sigma(-2) = 0.119.$

Sentence 4: $\hat{y} = \sigma(1 + 2 \cdot 0 - 4) = \sigma(-2) = 0.119.$



Logistic regression [Star War Example]

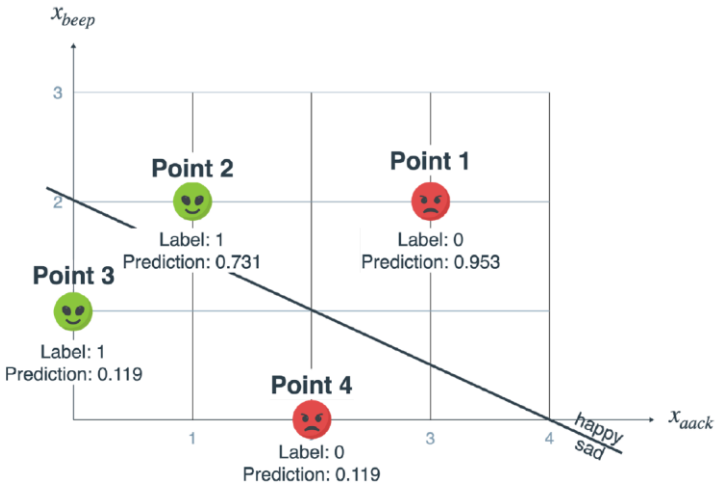
Prediction: $\hat{y} = \sigma(1 \cdot x_{aack} + 2 \cdot x_{beep} - 4)$

Sentence 1: $\hat{y} = \sigma(3 + 2 \cdot 2 - 4) = \sigma(3) = 0.953.$

Sentence 2: $\hat{y} = \sigma(1 + 2 \cdot 2 - 4) = \sigma(1) = 0.731.$

Sentence 3: $\hat{y} = \sigma(0 + 2 \cdot 1 - 4) = \sigma(-2) = 0.119.$

Sentence 4: $\hat{y} = \sigma(1 + 2 \cdot 0 - 4) = \sigma(-2) = 0.119.$



- **Point 1:**
  - Label = 0 (sad)
  - Prediction (probability of being happy) = 0.953
  - Probability of being sad: 1-0.953 = **0.047**

Logistic regression [**Star War Example**]

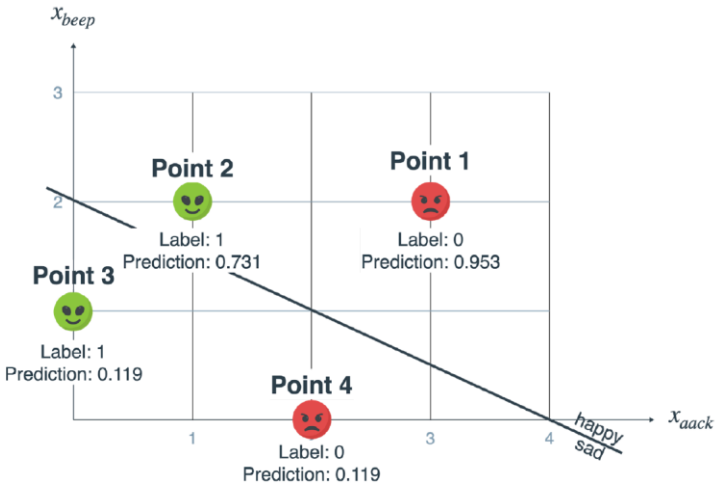
Prediction: $\hat{y} = \sigma(1 \cdot x_{aack} + 2 \cdot x_{beep} - 4)$

Sentence 1: $\hat{y} = \sigma(3 + 2 \cdot 2 - 4) = \sigma(3) = 0.953.$

Sentence 2: $\hat{y} = \sigma(1 + 2 \cdot 2 - 4) = \sigma(1) = 0.731.$

Sentence 3: $\hat{y} = \sigma(0 + 2 \cdot 1 - 4) = \sigma(-2) = 0.119.$

Sentence 4: $\hat{y} = \sigma(1 + 2 \cdot 0 - 4) = \sigma(-2) = 0.119.$



- **Point 1:**
  - Label = 0 (sad)
  - Prediction (probability of being happy) = 0.953
  - Probability of being sad: 1-0.953 = **0.047**
- **Point 2:**
  - Label = 1 (happy)
  - Prediction (probability of being happy) = 0.731
  - Probability of being happy: **0.731**

Logistic regression [Star War Example]

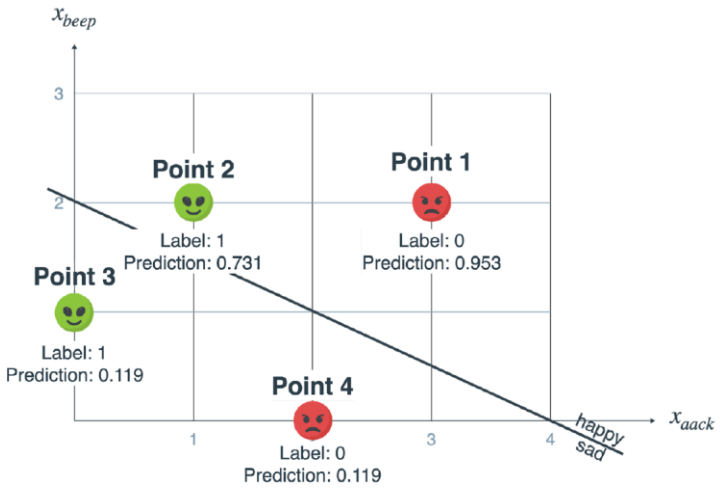
Prediction: $\hat{y} = \sigma(1 \cdot x_{aack} + 2 \cdot x_{beep} - 4)$

Sentence 1: $\hat{y} = \sigma(3 + 2 \cdot 2 - 4) = \sigma(3) = 0.953.$

Sentence 2: $\hat{y} = \sigma(1 + 2 \cdot 2 - 4) = \sigma(1) = 0.731.$

Sentence 3: $\hat{y} = \sigma(0 + 2 \cdot 1 - 4) = \sigma(-2) = 0.119.$

Sentence 4: $\hat{y} = \sigma(1 + 2 \cdot 0 - 4) = \sigma(-2) = 0.119.$



- **Point 1:**
  - Label = 0 (sad)
  - Prediction (probability of being happy) = 0.953
  - Probability of being sad: 1-0.953 = **0.047**
- **Point 2:**
  - Label = 1 (happy)
  - Prediction (probability of being happy) = 0.731
  - Probability of being happy: **0.731**
- **Point 3:**
  - Label = 1 (happy)
  - Prediction (probability of being happy) = 0.119
  - Probability of being happy: **0.119**

Logistic regression [Star War Example]

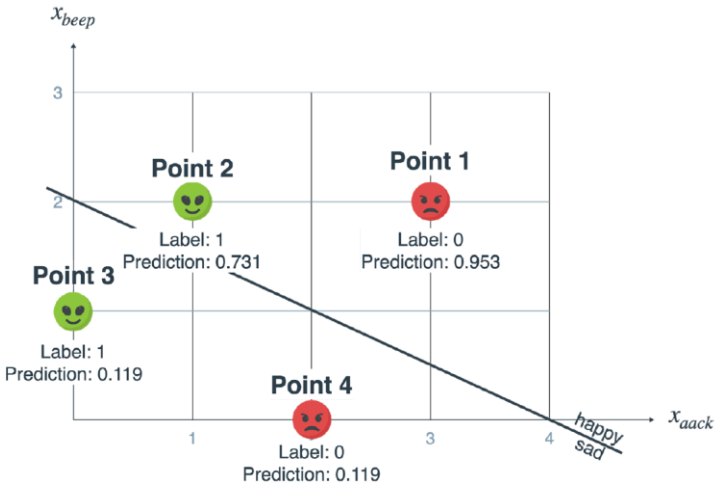
Prediction: $\hat{y} = \sigma(1 \cdot x_{aack} + 2 \cdot x_{beep} - 4)$

Sentence 1: $\hat{y} = \sigma(3 + 2 \cdot 2 - 4) = \sigma(3) = 0.953.$

Sentence 2: $\hat{y} = \sigma(1 + 2 \cdot 2 - 4) = \sigma(1) = 0.731.$

Sentence 3: $\hat{y} = \sigma(0 + 2 \cdot 1 - 4) = \sigma(-2) = 0.119.$

Sentence 4: $\hat{y} = \sigma(1 + 2 \cdot 0 - 4) = \sigma(-2) = 0.119.$



- **Point 1:**
  - Label = 0 (sad)
  - Prediction (probability of being happy) = 0.953
  - Probability of being sad: 1-0.953 = **0.047**
- **Point 2:**
  - Label = 1 (happy)
  - Prediction (probability of being happy) = 0.731
  - Probability of being happy: **0.731**
- **Point 3:**
  - Label = 1 (happy)
  - Prediction (probability of being happy) = 0.119
  - Probability of being happy: **0.119**
- **Point 4:**
  - Label = 0 (sad)
  - Prediction (probability of being happy) = 0.119
  - Probability of being sad: 1 - 0.119 = **0.881**

Lecture Overview

- Throwback
- Linear Regression As A Classifier!
- Problem Definition
- Perceptron Classifier [Formula]
- Perceptron Classifier [error function]
- Perceptron Classifier [Training]
- Perceptron Classifier [SKlearn]
- Logistic regression vs Perceptron Classifier
- Logistic regression [cross entropy error]
- Logistic regression [Training]
- Logistic regression [SKlearn]
- Logistic regression [Multi-class Model]

Logistic regression [**Log Loss (cross entropy error function)**]

Logistic regression [**Log Loss (cross entropy error function)**]

- As we deal with probability we needs the multiplication of predictions.

Logistic regression [**Log Loss (cross entropy error function)**]

- As we deal with probability we needs the multiplication of predictions.

$$0.047 \cdot 0.731 \cdot 0.119 \cdot 0.881 = 0.004$$

Logistic regression [**Log Loss (cross entropy error function)**]

- As we deal with probability we needs the multiplication of predictions.

$$0.047 \cdot 0.731 \cdot 0.119 \cdot 0.881 = 0.004$$

- The problem with multiplication is that:
  - The output might be very small as the number of predictions incases
  - The multiplication is hard to be done for that small number
- We have a way to transform multiplication to sum.. It is **Log_e**

Logistic regression [**Log Loss (cross entropy error function)**]

- As we deal with probability we needs the multiplication of predictions.

$$0.047 \cdot 0.731 \cdot 0.119 \cdot 0.881 = 0.004$$

- The problem with multiplication is that:
  - The output might be very small as the number of predictions incases
  - The multiplication is hard to be done for that small number
- We have a way to transform multiplication to sum.. It is **Log_e**

$$\begin{aligned} &ln\,(0.047 \cdot 0.731 \cdot 0.119 \cdot 0.881) \\ &\quad = ln\,(0.047) + ln\,(0.731) + ln\,(0.119) + ln\,(0.881) \\ &\quad = -5.616. \end{aligned}$$

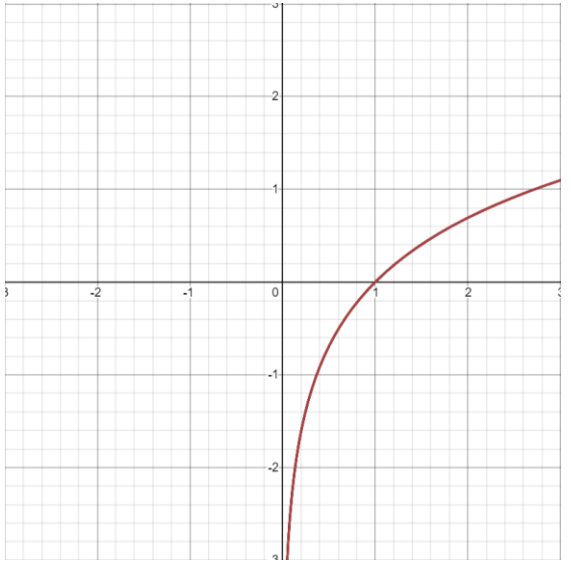
Logistic regression [**Log Loss (cross entropy error function)**]

- As we deal with probability we needs the multiplication of predictions.

$$0.047 \cdot 0.731 \cdot 0.119 \cdot 0.881 = 0.004$$

- The problem with multiplication is that:
  - The output might be very small as the number of predictions incases
  - The multiplication is hard to be done for that small number
- We have a way to transform multiplication to sum.. It is **Log_e**

$$\begin{aligned} &ln\,(0.047 \cdot 0.731 \cdot 0.119 \cdot 0.881) \\ &\quad = ln\,(0.047) + ln\,(0.731) + ln\,(0.119) + ln\,(0.881) \\ &\quad = -5.616. \end{aligned}$$



Logistic regression [**Log Loss (cross entropy error function)**]

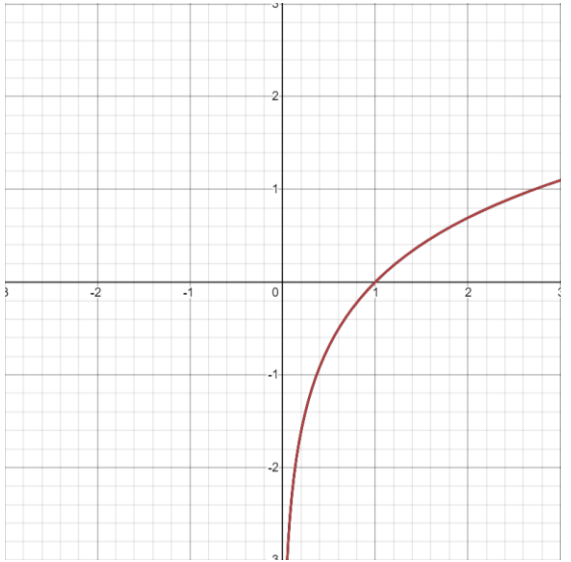
- As we deal with probability we needs the multiplication of predictions.

$$0.047 \cdot 0.731 \cdot 0.119 \cdot 0.881 = 0.004$$

- The problem with multiplication is that:
  - The output might be very small as the number of predictions incases
  - The multiplication is hard to be done for that small number
- We have a way to transform multiplication to sum.. It is **Log_e**

$$\begin{aligned} &ln\,(0.047 \cdot 0.731 \cdot 0.119 \cdot 0.881) \\ &\quad = ln\,(0.047) + ln\,(0.731) + ln\,(0.119) + ln\,(0.881) \\ &\quad = -5.616. \end{aligned}$$

- As you can see loge for values [0,1] is always negative so need to multiply the value by (-) to have the error.
- As the input of **Log_e** be near to 1 the loss decreased



Logistic regression [**Log Loss (cross entropy error function)**]

Logistic regression [**Log Loss (cross entropy error function)**]

If the label is 0:
Log loss = -ln(1-y')

If the label is 1:
Log loss = -ln(y')

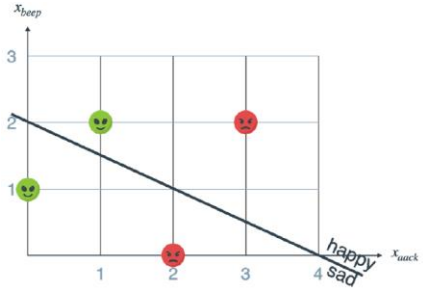
We can combine the 2 cases in this formula:
Log loss = -y ln(y') - (1-y) ln (1-y')

Logistic regression [Log Loss (cross entropy error function)]

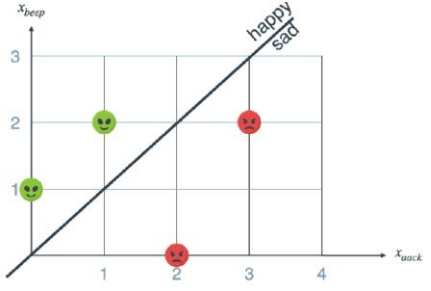
If the label is 0:
Log loss = -ln(1-y')

If the label is 1:
Log loss = -ln(y')

We can combine the 2 cases in this formula:
Log loss = -y ln(y') - (1-y) ln (1-y')



Classifier 1



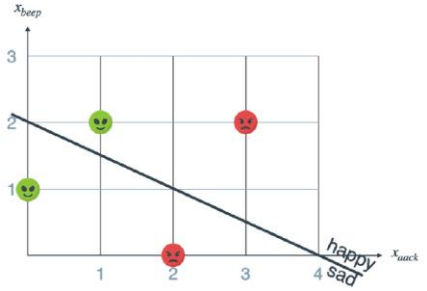
Classifier 2

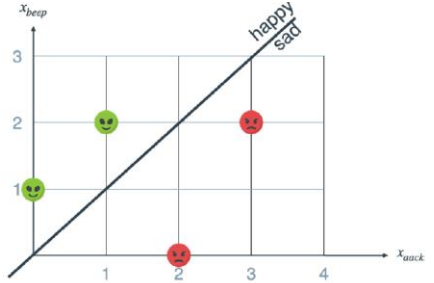
Logistic regression [Log Loss (cross entropy error function)]

If the label is 0:
Log loss = -ln(1-y')

If the label is 1:
Log loss = -ln(y')

We can combine the 2 cases in this formula:
Log loss = -y ln(y') - (1-y) ln (1-y')





Classifier 1

Classifier 2

Point	Label	Classifier 1 prediction	Classifier 2 prediction
1	0 (Sad)	0.953	0.269
2	1 (Happy)	0.731	0.731
3	1 (Happy)	0.119	0.731
4	0 (Sad)	0.881	0.119

Logistic regression [Log Loss (cross entropy error function)]

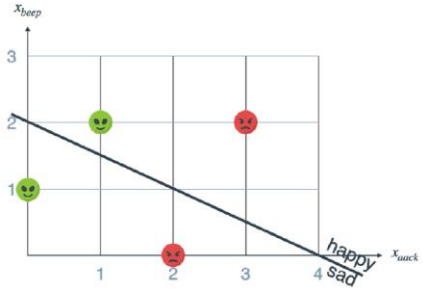
If the label is 0:
Log loss = -ln(1-y')

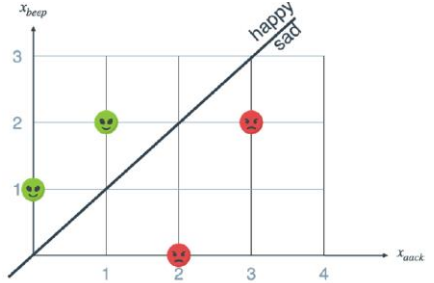
If the label is 1:
Log loss = -ln(y')

We can combine the 2 cases in this formula:
Log loss = -y ln(y') - (1-y) ln (1-y')

Classifier 1 Log Loss: 5.616

Classifier 2 Log Loss: 1.067





Classifier 1 Classifier 2

Point	Label	Classifier 1 prediction	Classifier 2 prediction
1	0 (Sad)	0.953	0.269
2	1 (Happy)	0.731	0.731
3	1 (Happy)	0.119	0.731
4	0 (Sad)	0.881	0.119

Logistic regression [Log Loss (cross entropy error function)]

If the label is 0:
Log loss = -ln(1-y')

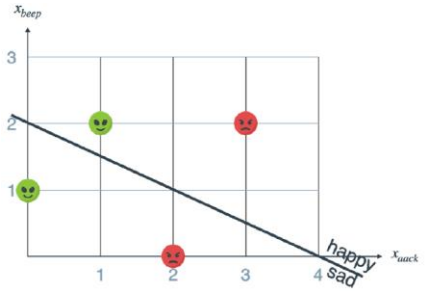
If the label is 1:
Log loss = -ln(y')

We can combine the 2 cases in this formula:
Log loss = -y ln(y') - (1-y) ln (1-y')

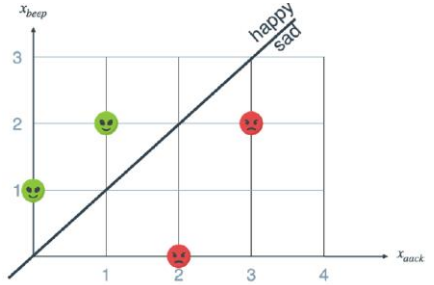
Classifier 1 Log Loss: 5.616

Classifier 2 Log Loss: 1.067

What do we need now?



Classifier 1



Classifier 2

Point	Label	Classifier 1 prediction	Classifier 2 prediction
1	0 (Sad)	0.953	0.269
2	1 (Happy)	0.731	0.731
3	1 (Happy)	0.119	0.731
4	0 (Sad)	0.881	0.119

Prediction and Error Function

Use colab to open this github notebook:

“s7s/machine_learning_1/logistic_regression/Coding_logistic_regression.ipynb”

Lecture Overview

- Throwback
- Linear Regression As A Classifier!
- Problem Definition
- Perceptron Classifier [Formula]
- Perceptron Classifier [error function]
- Perceptron Classifier [Training]
- Perceptron Classifier [SKlearn]
- Logistic regression vs Perceptron Classifier
- Logistic regression [cross entropy error]
- Logistic regression [Training]
- Logistic regression [SKlearn]
- Logistic regression [Multi-class Model]

Logistic regression [**Training**]

Logistic regression [Training]

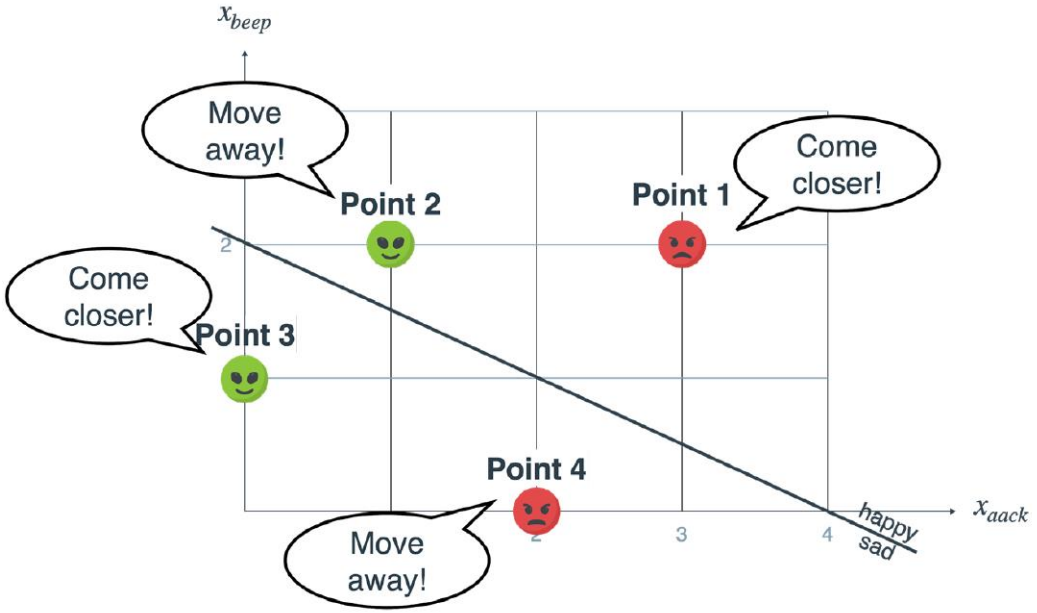
For **Perceptron classifier** it was:
• Case 1: If the point is correctly classified, leave the line as it is.
• Case 2: If the point is incorrectly classified, move the line a little closer to the point.

But for our **Logistic classifier**:
• Case 1: If the point is **correctly** classified, move the line a little **away** to the point..
• Case 2: If the point is **incorrectly** classified, move the line a little **closer** to the point.

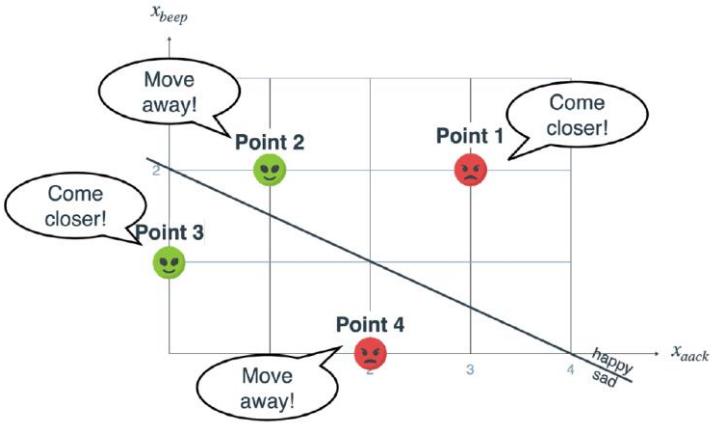
Logistic regression [Training]

For **Perceptron classifier** it was:
• Case 1: If the point is correctly classified, leave the line as it is.
• Case 2: If the point is incorrectly classified, move the line a little closer to the point.

But for our **Logistic classifier**:
• Case 1: If the point is **correctly** classified, move the line a little **away** to the point..
• Case 2: If the point is **incorrectly** classified, move the line a little **closer** to the point.

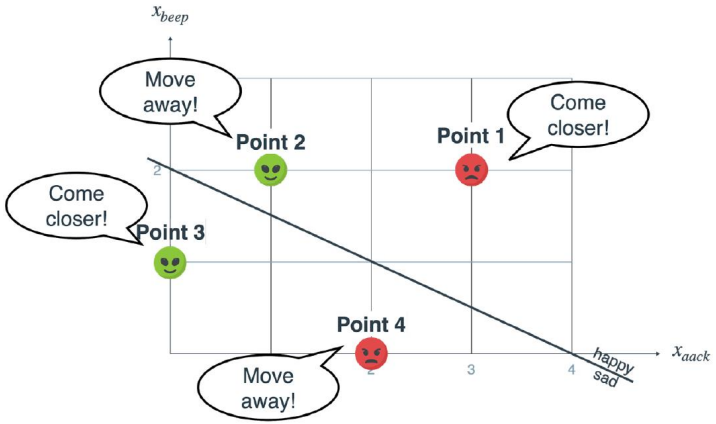


Logistic regression [**Training**]



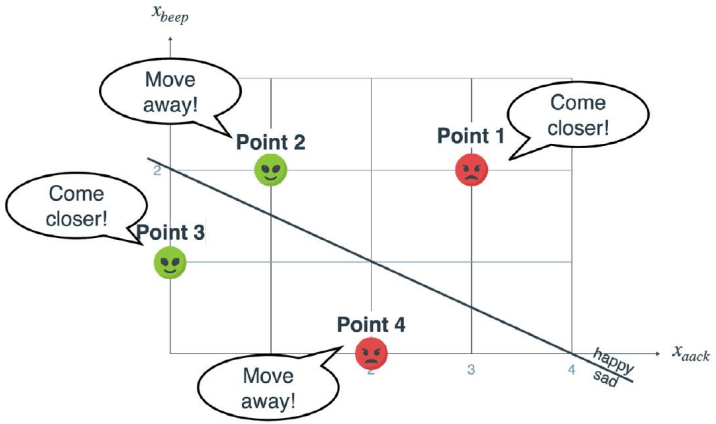
Logistic regression [Training]

Point	Label y	Classifier 1 prediction $\hat{y}$	How to tune the weights a, b, and the bias c	$y - \hat{y}$
1	0	0.953	Decrease by a large amount	-0.953
2	1	0.731	Increase by a small amount	0.269
3	1	0.119	Increase by a large amount	0.881
4	0	0.119	Decrease by a small amount	-0.119



Logistic regression [Training]

Point	Label y	Classifier 1 prediction $\hat{y}$	How to tune the weights a, b, and the bias c	$y - \hat{y}$
1	0	0.953	Decrease by a large amount	-0.953
2	1	0.731	Increase by a small amount	0.269
3	1	0.119	Increase by a large amount	0.881
4	0	0.119	Decrease by a small amount	-0.119

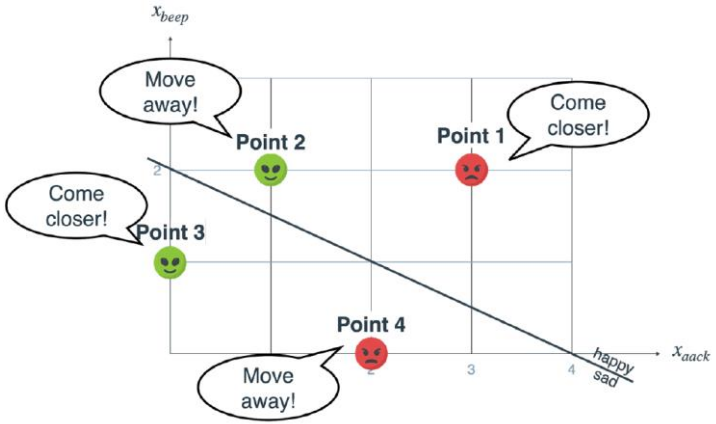


Observation: there is a similarity between the increase/decrease column and (y- y') column.

Base on this observation, we can modify the weights and bias with this formula:

Logistic regression [Training]

Point	Label y	Classifier 1 prediction $\hat{y}$	How to tune the weights a, b, and the bias c	$y - \hat{y}$
1	0	0.953	Decrease by a large amount	-0.953
2	1	0.731	Increase by a small amount	0.269
3	1	0.119	Increase by a large amount	0.881
4	0	0.119	Decrease by a small amount	-0.119



Observation: there is a similarity between the increase/decrease column and (y- y') column.

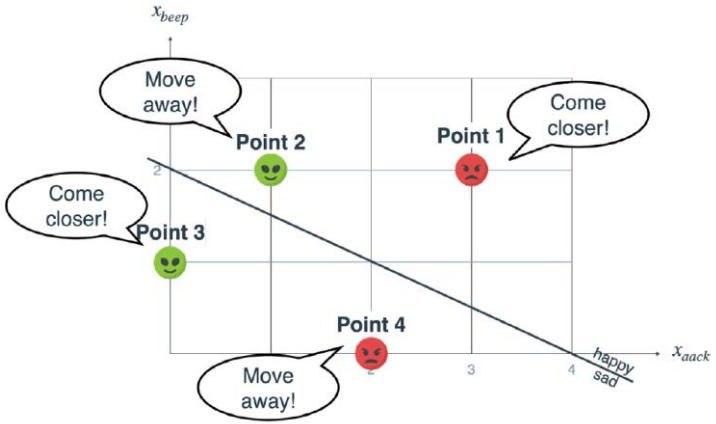
Base on this observation, we can modify the weights and bias with this formula:

- $a' = a + \eta(y - \hat{y})$
- $b' = b + \eta(y - \hat{y})$
- $c' = c + \eta(y - \hat{y})$.

Where y' = sigmoid($ax_1 + bx_2 + c$)

Logistic regression [Training]

Point	Label y	Classifier 1 prediction $\hat{y}$	How to tune the weights a, b, and the bias c	$y - \hat{y}$
1	0	0.953	Decrease by a large amount	-0.953
2	1	0.731	Increase by a small amount	0.269
3	1	0.119	Increase by a large amount	0.881
4	0	0.119	Decrease by a small amount	-0.119



Observation: there is a similarity between the increase/decrease column and (y- y') column.

Base on this observation, we can modify the weights and bias with this formula:

- $a' = a + \eta(y - \hat{y})x_1$
- $b' = b + \eta(y - \hat{y})x_2$
- $c' = c + \eta(y - \hat{y})$.

Where y' = sigmoid($ax_1 + bx_2 + c$)

Logistic regression [Training]

Inputs:

- A logistic classifier with weights $a, b,$ and bias $c.$
- A point with coordinates (x_1, x_2) and label $y.$
- A small value η (the learning rate).

Output:

- A perceptron with new weights $a', b',$ and bias c' which is at least as good as the input perceptron for that point.

Procedure:

- The prediction the perceptron makes at the point is $\hat{y} = \sigma(ax_1 + bx_2 + c).$
- **Return** the perceptron with the following weights and bias:
  - $a' = a + \eta(y - \hat{y})x_1$
  - $b' = b + \eta(y - \hat{y})x_2$
  - $c' = c + \eta(y - \hat{y}).$

Mathematically:

loss function, the Cross Entropy Loss function.

$$\mathcal{L} = -\frac{1}{N}\sum_i y_i \log(a_i) + (1 - y_i)\log(1 - a_i)$$

Logistic regression [Training]

Inputs:

- A logistic classifier with weights $a, b,$ and bias $c.$
- A point with coordinates (x_1, x_2) and label $y.$
- A small value η (the learning rate).

Output:

- A perceptron with new weights $a', b',$ and bias c' which is at least as good as the input perceptron for that point.

Procedure:

- The prediction the perceptron makes at the point is $\hat{y} = \sigma(ax_1 + bx_2 + c).$
- **Return** the perceptron with the following weights and bias:
  - $a' = a + \eta(y - \hat{y})x_1$
  - $b' = b + \eta(y - \hat{y})x_2$
  - $c' = c + \eta(y - \hat{y}).$

Mathematically:

loss function, the Cross Entropy Loss function.

$$\mathcal{L} = -\frac{1}{N}\sum_i y_i \log(a_i) + (1 - y_i)\log(1 - a_i)$$

.

The following remains the same:

$$z_i = W x_i$$
$$a_i = \sigma(z_i)$$

Logistic regression [Training]

Inputs:

- A logistic classifier with weights $a, b,$ and bias $c.$
- A point with coordinates (x_1, x_2) and label $y.$
- A small value η (the learning rate).

Output:

- A perceptron with new weights $a', b',$ and bias c' which is at least as good as the input perceptron for that point.

Procedure:

- The prediction the perceptron makes at the point is $\hat{y} = \sigma(ax_1 + bx_2 + c).$
- **Return** the perceptron with the following weights and bias:
  - $a' = a + \eta(y - \hat{y})x_1$
  - $b' = b + \eta(y - \hat{y})x_2$
  - $c' = c + \eta(y - \hat{y}).$

Mathematically:

loss function, the Cross Entropy Loss function.

$$\mathcal{L} = -\frac{1}{N}\sum_i y_i \log(a_i) + (1 - y_i)\log(1 - a_i)$$

.

The following remains the same:

$$z_i = W x_i$$
$$a_i = \sigma(z_i)$$

and the derivative,

$$\frac{d\mathcal{L}}{dW} = \sum_i \frac{d\mathcal{L}}{da_i}\frac{da_i}{dz_i}\frac{dz_i}{dW}$$

Logistic regression [Training]

Inputs:

- A logistic classifier with weights $a, b,$ and bias $c.$
- A point with coordinates (x_1, x_2) and label $y.$
- A small value η (the learning rate).

Output:

- A perceptron with new weights $a', b',$ and bias c' which is at least as good as the input perceptron for that point.

Procedure:

- The prediction the perceptron makes at the point is $\hat{y} = \sigma(ax_1 + bx_2 + c).$
- **Return** the perceptron with the following weights and bias:
  - $a' = a + \eta(y - \hat{y})x_1$
  - $b' = b + \eta(y - \hat{y})x_2$
  - $c' = c + \eta(y - \hat{y}).$

Mathematically:

loss function, the Cross Entropy Loss function.

$$\mathcal{L} = -\frac{1}{N}\sum_i y_i \log(a_i) + (1 - y_i)\log(1 - a_i)$$

.

The following remains the same:

$$z_i = W x_i$$
$$a_i = \sigma(z_i)$$

and the derivative,

$$\frac{d\mathcal{L}}{dW} = \sum_i \frac{d\mathcal{L}}{da_i}\frac{da_i}{dz_i}\frac{dz_i}{dW}$$
$$= \frac{1}{N}\sum_i -\left(\frac{y_i}{a_i} - \frac{1 - y_i}{1 - a_i}\right) \cdot a_i(1 - a_i) \cdot x_i$$

Logistic regression [Training]

Inputs:

- A logistic classifier with weights $a, b,$ and bias $c.$
- A point with coordinates (x_1, x_2) and label $y.$
- A small value η (the learning rate).

Output:

- A perceptron with new weights $a', b',$ and bias c' which is at least as good as the input perceptron for that point.

Procedure:

- The prediction the perceptron makes at the point is $\hat{y} = \sigma(ax_1 + bx_2 + c).$
- **Return** the perceptron with the following weights and bias:
  - $a' = a + \eta(y - \hat{y})x_1$
  - $b' = b + \eta(y - \hat{y})x_2$
  - $c' = c + \eta(y - \hat{y}).$

Mathematically:

loss function, the Cross Entropy Loss function.

$$\mathcal{L} = -\frac{1}{N}\sum_i y_i \log(a_i) + (1 - y_i)\log(1 - a_i)$$

.

The following remains the same:

$$z_i = W x_i$$
$$a_i = \sigma(z_i)$$

and the derivative,

$$\frac{d\mathcal{L}}{dW} = \sum_i \frac{d\mathcal{L}}{da_i}\frac{da_i}{dz_i}\frac{dz_i}{dW}$$
$$= \frac{1}{N}\sum_i -\left(\frac{y_i}{a_i} - \frac{1 - y_i}{1 - a_i}\right) \cdot a_i(1 - a_i) \cdot x_i$$
$$= \frac{1}{N}\sum_i -\left(\frac{y_i - a_i}{a_i(1 - a_i)}\right) \cdot a_i(1 - a_i) \cdot x_i$$

Logistic regression [Training]

Inputs:

- A logistic classifier with weights $a, b,$ and bias $c.$
- A point with coordinates (x_1, x_2) and label $y.$
- A small value η (the learning rate).

Output:

- A perceptron with new weights $a', b',$ and bias c' which is at least as good as the input perceptron for that point.

Procedure:

- The prediction the perceptron makes at the point is $\hat{y} = \sigma(ax_1 + bx_2 + c).$
- **Return** the perceptron with the following weights and bias:
  - $a' = a + \eta(y - \hat{y})x_1$
  - $b' = b + \eta(y - \hat{y})x_2$
  - $c' = c + \eta(y - \hat{y}).$

Mathematically:

loss function, the Cross Entropy Loss function.

$$\mathcal{L} = -\frac{1}{N}\sum_i y_i \log(a_i) + (1 - y_i)\log(1 - a_i)$$

.

The following remains the same:

$$z_i = W x_i$$
$$a_i = \sigma(z_i)$$

and the derivative,

$$\frac{d\mathcal{L}}{dW} = \sum_i \frac{d\mathcal{L}}{da_i}\frac{da_i}{dz_i}\frac{dz_i}{dW}$$
$$= \frac{1}{N}\sum_i -\left(\frac{y_i}{a_i} - \frac{1 - y_i}{1 - a_i}\right) \cdot a_i(1 - a_i) \cdot x_i$$
$$= \frac{1}{N}\sum_i -\left(\frac{y_i - a_i}{a_i(1 - a_i)}\right) \cdot a_i(1 - a_i) \cdot x_i$$
$$= \frac{1}{N}\sum_i -(y_i - a_i) \cdot x_i$$

Logistic regression [Training]

Inputs:

- A logistic classifier with weights $a, b,$ and bias $c.$
- A point with coordinates (x_1, x_2) and label $y.$
- A small value η (the learning rate).

Output:

- A perceptron with new weights $a', b',$ and bias c' which is at least as good as the input perceptron for that point.

Procedure:

- The prediction the perceptron makes at the point is $\hat{y} = \sigma(ax_1 + bx_2 + c).$
- **Return** the perceptron with the following weights and bias:
  - $a' = a + \eta(y - \hat{y})x_1$
  - $b' = b + \eta(y - \hat{y})x_2$
  - $c' = c + \eta(y - \hat{y}).$

Mathematically:

loss function, the Cross Entropy Loss function.

$$\mathcal{L} = -\frac{1}{N}\sum_i y_i \log(a_i) + (1 - y_i)\log(1 - a_i)$$

.

The following remains the same:

$$z_i = W x_i$$
$$a_i = \sigma(z_i)$$

and the derivative,

$$\frac{d\mathcal{L}}{dW} = \sum_i \frac{d\mathcal{L}}{da_i}\frac{da_i}{dz_i}\frac{dz_i}{dW}$$
$$= \frac{1}{N}\sum_i -\left(\frac{y_i}{a_i} - \frac{1 - y_i}{1 - a_i}\right) \cdot a_i(1 - a_i) \cdot x_i$$
$$= \frac{1}{N}\sum_i -\left(\frac{y_i - a_i}{a_i(1 - a_i)}\right) \cdot a_i(1 - a_i) \cdot x_i$$
$$= \frac{1}{N}\sum_i -(y_i - a_i) \cdot x_i$$

w'=w - n dE/dw

Logistic regression [Training]

Inputs:

- A logistic classifier with weights $a, b,$ and bias $c.$
- A point with coordinates (x_1, x_2) and label $y.$
- A small value η (the learning rate).

Output:

- A perceptron with new weights $a', b',$ and bias c' which is at least as good as the input perceptron for that point.

Procedure:

- The prediction the perceptron makes at the point is $\hat{y} = \sigma(ax_1 + bx_2 + c).$
- **Return** the perceptron with the following weights and bias:
  - $a' = a + \eta(y - \hat{y})x_1$
  - $b' = b + \eta(y - \hat{y})x_2$
  - $c' = c + \eta(y - \hat{y}).$

Mathematically:

loss function, the Cross Entropy Loss function.

$$\mathcal{L} = -\frac{1}{N}\sum_i y_i \log(a_i) + (1 - y_i)\log(1 - a_i)$$

.

The following remains the same:

$$z_i = W x_i$$
$$a_i = \sigma(z_i)$$

and the derivative,

$$\frac{d\mathcal{L}}{dW} = \sum_i \frac{d\mathcal{L}}{da_i}\frac{da_i}{dz_i}\frac{dz_i}{dW}$$
$$= \frac{1}{N}\sum_i -\left(\frac{y_i}{a_i} - \frac{1 - y_i}{1 - a_i}\right) \cdot a_i(1 - a_i) \cdot x_i$$
$$= \frac{1}{N}\sum_i -\left(\frac{y_i - a_i}{a_i(1 - a_i)}\right) \cdot a_i(1 - a_i) \cdot x_i$$
$$= \frac{1}{N}\sum_i -(y_i - a_i) \cdot x_i$$

$w' = w - \eta\, \partial E/\partial w$

$w' = w + \eta\, (1/N)\, (y - y')x_i$

Weights update using one point

Use colab to open this github notebook:

“s7s/machine_learning_1/logistic_regression/Coding_logistic_regression.ipynb”

Lecture Overview

- Throwback
- Linear Regression As A Classifier!
- Problem Definition
- Perceptron Classifier [Formula]
- Perceptron Classifier [error function]
- Perceptron Classifier [Training]
- Perceptron Classifier [SKlearn]
- Logistic regression vs Perceptron Classifier
- Logistic regression [cross entropy error]
- Logistic regression [Training]
- Logistic regression [SKlearn]
- Logistic regression [Multi-class Model]

SKlearn

Use colab to open this github notebook:

“s7s/machine_learning_1/logistic_regression/Sentiment_analysis_IMDB.ipynb”

SKlearn

```
>>> from sklearn.datasets import load_iris
>>> from sklearn.linear_model import LogisticRegression
>>> X, y = load_iris(return_X_y=True)
>>> clf = LogisticRegression(random_state=0).fit(X, y)
>>> clf.predict(X[:2, :])
array([0, 0])
>>> clf.predict_proba(X[:2, :])
array([[9.8...e-01, 1.8...e-02, 1.4...e-08],
       [9.7...e-01, 2.8...e-02, ...e-08]])
>>> clf.score(X, y)
0.97...
```

Lecture Overview

- Throwback
- Linear Regression As A Classifier!
- Problem Definition
- Perceptron Classifier [Formula]
- Perceptron Classifier [error function]
- Perceptron Classifier [Training]
- Perceptron Classifier [SKlearn]
- Logistic regression vs Perceptron Classifier
- Logistic regression [cross entropy error]
- Logistic regression [Training]
- Logistic regression [SKlearn]
- Logistic regression [Multi-class Model]

Multi-class Model

- We can use one-to-many method but we need to find a method to get the probability of each class.
- To get the probabilities we use the **softmax** function (general form of sigmoid).
- For the output of $ax_1 + bx_2 + c$, let's imagine these results:
  Dog Classifier: 3 Cat Classifier: 2 Bird Classifier: -1
- We can use this result over the sum but Bird will have a negative probability -¼,
  so we need a function that is always positive, we can use e^x as it is also has a simple derivative.
- Dog Classifier: e^3= 20.085 Cat Classifier: e^2 =7.389 Bird Classifier: e^{-1}=0.368
- Probability of dog: Probability of cat: Probability of bird:
  20.085/27.842 = 0.721 7.389/27.842 = 0.265 0.368/27.842 = 0.013
- General formula of **Softmax**:

Multi-class Model

- We can use one-to-many method but we need to find a method to get the probability of each class.
- To get the probabilities we use the **softmax** function (general form of sigmoid).
- For the output of $ax_1 + bx_2 + c$, let's imagine these results:
  Dog Classifier: 3 Cat Classifier: 2 Bird Classifier: -1
- We can use this result over the sum but Bird will have a negative probability -¼,
  so we need a function that is always positive, we can use e^x as it is also has a simple derivative.
- Dog Classifier: e^3= 20.085 Cat Classifier: e^2 =7.389 Bird Classifier: e^{-1}=0.368
- Probability of dog: Probability of cat: Probability of bird:
  20.085/27.842 = 0.721 7.389/27.842 = 0.265 0.368/27.842 = 0.013
- General formula of **Softmax**:

$$p_i = \frac{e^{a_i}}{e^{a_1} + e^{a_2} + \cdots + e^{a_n}}$$

Lecture Overview

- Throwback
- Linear Regression As A Classifier!
- Problem Definition
- Perceptron Classifier [Formula]
- Perceptron Classifier [error function]
- Perceptron Classifier [Training]
- Perceptron Classifier [SKlearn]
- Logistic regression vs Perceptron Classifier
- Logistic regression [cross entropy error]
- Logistic regression [Training]
- Logistic regression [SKlearn]
- Logistic regression [Multi-class Model]

Feedback