How do Machines Learn?

COM 100π – Advanced Presentational Speaking

J. Setpal

February 28, 2024

Linear Regression

Let's start with the definition of a line.

$$y = mx + b$$

Linear Regression

Let's start with the definition of a line.

$$y = mx + b$$

Using just these two sliders $(m,\ b)$, I can make any line on the 2D plane!

Linear Regression

Let's start with the definition of a line.

$$y = mx + b$$

Using just these two sliders $(m,\ b)$, I can make any line on the 2D plane!

We can use this to demonstrate a linear relationship between an input (x) an output (y).

Let's start with the definition of a line.

$$y = mx + b$$

Using just these two sliders $(m,\ b)$, I can make any line on the 2D plane!

We can use this to demonstrate a linear relationship between an input (x) an output (y). Assume we have the following data:

x	**y**
1	2
2	3
3	4
4	5

Linear Regression

Let's start with the definition of a line.

$$y = mx + b$$

Using just these two sliders $(m,\ b)$, I can make any line on the 2D plane!

We can use this to demonstrate a linear relationship between an input (x) an output (y). Assume we have the following data:

x	**y**
1	2
2	3
3	4
4	5

(x, y) is mapped by $y = 1 \cdot x + 1$

Linear Regression

Let's start with the definition of a line.

$$y = mx + b$$

Using just these two sliders $(m,\ b)$, I can make any line on the 2D plane!

We can use this to demonstrate a linear relationship between an input (x) an output (y). Assume we have the following data:

x	**y**
1	2
2	3
3	4
4	5

(x, y) is mapped by $y = 1 \cdot x + 1$

This allows us to represent 4 data points with 2 numbers.

Linear Regression

Let's start with the definition of a line.

$$y = mx + b$$

Using just these two sliders $(m,\ b)$, I can make any line on the 2D plane!

We can use this to demonstrate a linear relationship between an input (x) an output (y). Assume we have the following data:

x	**y**
1	2
2	3
3	4
4	5

(x, y) is mapped by $y = 1 \cdot x + 1$

This allows us to represent 4 data points with 2 numbers.

It quantifies the essence of the data *without memorizing it*; so, is **machine learning**.

Loss Function

Now, what do we do if our data is weird[1]?

x	y
1	1.8
2	3.2
3	4.1
4	6

[1]weird = not strictly linear

Loss Function

Now, what do we do if our data is weird[1]?

x	y
1	1.8
2	3.2
3	4.1
4	6

We cannot find a pair of weights that satisfies all our input-output pairs (the dataset).

[1] weird = not strictly linear

Loss Function

Now, what do we do if our data is weird[1]?

x	y
1	1.8
2	3.2
3	4.1
4	6

We cannot find a pair of weights that satisfies all our input-output pairs (the dataset).

However, we can still find a line that *approximately* satisfies the data. It's called the **line of best fit**.

[1] weird = not strictly linear

Loss Function

Now, what do we do if our data is weird[1]?

x	**y**
1	1.8
2	3.2
3	4.1
4	6

We cannot find a pair of weights that satisfies all our input-output pairs (the dataset).

However, we can still find a line that *approximately* satisfies the data. It's called the **line of best fit**.

So; how would we find this line?

[1]weird = not strictly linear

Loss Function

Now, what do we do if our data is weird[1]?

x	y
1	1.8
2	3.2
3	4.1
4	6

We cannot find a pair of weights that satisfies all our input-output pairs (the dataset).

However, we can still find a line that *approximately* satisfies the data. It's called the **line of best fit**.

So; how would we find this line? We create a differentiable function we can use as the **score** for the line.

[1]weird = not strictly linear

Loss Function

Now, what do we do if our data is weird[1]?

x	y
1	1.8
2	3.2
3	4.1
4	6

We cannot find a pair of weights that satisfies all our input-output pairs (the dataset).

However, we can still find a line that *approximately* satisfies the data. It's called the **line of best fit**.

So; how would we find this line? We create a differentiable function we can use as the **score** for the line.

Whichever line has the best score, is the line of best fit. For Linear Regression, we can use **Mean Squared Error**:

$$L(y_{\text{pred}}, y_{\text{actual}}) = (y_{\text{actual}} - y_{\text{pred}})^2$$

[1] weird = not strictly linear

Activation Functions

Now, what do we do if our data is categorical?

x	**y**
1	0
2	0
3	1
4	1

Activation Functions

Now, what do we do if our data is categorical?

We can subvert the linear paradigm we saw before.

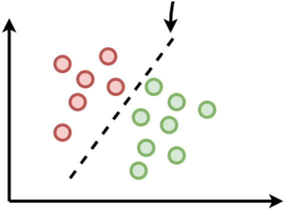
x	y
1	0
2	0
3	1
4	1

Activation Functions

Now, what do we do if our data is categorical?

x	**y**
1	0
2	0
3	1
4	1

We can subvert the linear paradigm we saw before. Instead of y denoting the output, it denotes a **decision boundary**.



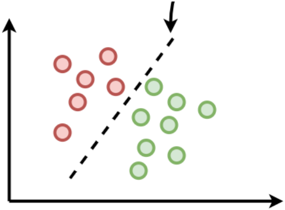
Everything *under* the line belongs to the **green** class, while everything *over* the line belongs to the **red** class.

Activation Functions

Now, what do we do if our data is categorical?

x	y
1	0
2	0
3	1
4	1

We can subvert the linear paradigm we saw before. Instead of y denoting the output, it denotes a **decision boundary**.



Everything *under* the line belongs to the **green** class, while everything *over* the line belongs to the **red** class.

Mathematically, we can simply map linear outputs using the sigmoid:

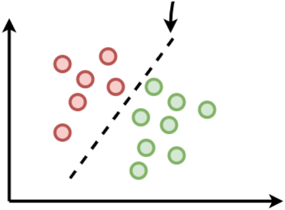
$$\sigma_{\text{sigmoid}}(x) = \frac{e^x}{1 + e^x}$$

Activation Functions

Now, what do we do if our data is categorical?

x	y
1	0
2	0
3	1
4	1

We can subvert the linear paradigm we saw before. Instead of y denoting the output, it denotes a **decision boundary**.



Everything *under* the line belongs to the **green** class, while everything *over* the line belongs to the **red** class.

Mathematically, we can simply map linear outputs using the sigmoid:

$$\sigma_{\text{sigmoid}}(x) = \frac{e^x}{1 + e^x}$$

This map called an **activation function**. Our new line function is:

$$y = \sigma_{\text{sigmoid}}(mx + b)$$

Multi-Layer Perceptron #1

This is where we bring it all togther.

Multi-Layer Perceptron #1

This is where we bring it all togther. What do we do if our *data* is not linearly separable?

x	**y**
1	1
2	0
3	0
4	1

Multi-Layer Perceptron #1

This is where we bring it all togther. What do we do if our *data* is not linearly separable?

x	**y**
1	1
2	0
3	0
4	1

Idea: Let's stack two linear predictors one after another.

$$y = m_2(m_1 x + b_1) + b_2$$

Multi-Layer Perceptron #1

This is where we bring it all togther. What do we do if our *data* is not linearly separable?

x	**y**
1	1
2	0
3	0
4	1

Idea: Let's stack two linear predictors one after another.

$$y = m_2(m_1 x + b_1) + b_2$$

This will allow the decision boundary to be *non-linear*! The architecture is called a **Multi-Layer Perceptron**.

Multi-Layer Perceptron #1

This is where we bring it all togther. What do we do if our *data* is not linearly separable?

x	y
1	1
2	0
3	0
4	1

Idea: Let's stack two linear predictors one after another.

$$y = m_2(m_1 x + b_1) + b_2$$

This will allow the decision boundary to be *non-linear*!
The architecture is called a **Multi-Layer Perceptron**.

However, there is a problem. The above equation can be reduced down to a single linear function.

Multi-Layer Perceptron #1

This is where we bring it all togther. What do we do if our *data* is not linearly separable?

x	**y**
1	1
2	0
3	0
4	1

Idea: Let's stack two linear predictors one after another.

$$y = m_2(m_1 x + b_1) + b_2$$

This will allow the decision boundary to be *non-linear*!
The architecture is called a **Multi-Layer Perceptron**.

However, there is a problem. The above equation can be reduced down to a single linear function.

We can solve this by adding **nonlinearity**, using the ReLU activation:

$$\sigma_{\text{relu}}(x) = max(0,\ x)$$

Multi-Layer Perceptron #2

Finally, we add the sigmoid activation to make the predictions class probabilities. Our new equation looks like:

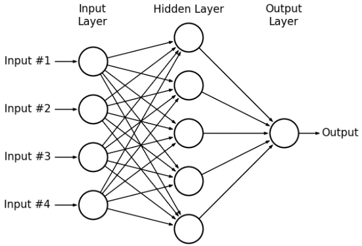
$$y = \sigma_{\text{sigmoid}}(m_2(\sigma_{\text{relu}}(m_1 x + b_1)) + b_2)$$

Multi-Layer Perceptron #2

Finally, we add the sigmoid activation to make the predictions class probabilities. Our new equation looks like:

$$y = \sigma_{\text{sigmoid}}(m_2(\sigma_{\text{relu}}(m_1 x + b_1)) + b_2)$$

Graphically, this is what it looks like:



Let's have some fun: `https://playground.tensorflow.org/`

Thank you!

Have an awesome rest of your day!

Slides:

https://www.cs.purdue.edu/homes/jsetpal/slides/how-ml.pdf