Gradient Descent

Towards Neural Networks

Justin Stevens

Outline

1. Decision Making
   - Perceptrons
   - Activation Functions
2. Classifying Digits through MNIST

Should I Stay or Should I Go?

Let's say I'm deciding on a given day whether or not to go to an Edmonton Oilers game. Here are the factors that will influence my decision:

Should I Stay or Should I Go?

Let's say I'm deciding on a given day whether or not to go to an Edmonton Oilers game. Here are the factors that will influence my decision:

- Are the tickets cheap or expensive?
- Do I have the time to go?
- Do I care about the team they're playing?

Should I Stay or Should I Go?

Let's say I'm deciding on a given day whether or not to go to an Edmonton Oilers game. Here are the factors that will influence my decision:

- Are the tickets cheap or expensive?
- Do I have the time to go?
- Do I care about the team they're playing?

We'll make my decision by encoding each possible input as a vector $\bar{x}$:

Should I Stay or Should I Go?

Let's say I'm deciding on a given day whether or not to go to an Edmonton Oilers game. Here are the factors that will influence my decision:

- Are the tickets cheap or expensive?
- Do I have the time to go?
- Do I care about the team they're playing?

We'll make my decision by encoding each possible input as a vector $\bar{x}$:

Ticket Prices	Availability	Interest	$\bar{x}$
Cheap	Yes	Yes	$(1, 1, 1)$
Cheap	No	No	$(1, 0, 0)$
Cheap	Yes	No	$(1, 1, 0)$
Cheap	No	Yes	$(1, 0, 1)$
Expensive	Yes	Yes	$(0, 1, 1)$
Expensive	No	No	$(0, 0, 0)$
Expensive	No	Yes	$(0, 0, 1)$
Expensive	Yes	No	$(0, 1, 0)$

How Will I Make my Decision?

Let's say I don't care much about price, but I do care about my availability and interest. In this case, the corresponding weights might be $\bar{w} = (1, 6, 3)$.

How Will I Make my Decision?

Let's say I don't care much about price, but I do care about my availability and interest. In this case, the corresponding weights might be $\bar{w} = (1, 6, 3)$. We can then compute the dot product $\bar{w} \cdot \bar{x}$ for each possible input:

How Will I Make my Decision?

Let's say I don't care much about price, but I do care about my availability and interest. In this case, the corresponding weights might be $\bar{w} = (1, 6, 3)$. We can then compute the dot product $\bar{w} \cdot \bar{x}$ for each possible input:

Ticket Prices	Availability	Interest	$\bar{x}$	$\bar{w} \cdot \bar{x}$
Cheap	Yes	Yes	$(1, 1, 1)$	10
Cheap	No	No	$(1, 0, 0)$	1
Cheap	Yes	No	$(1, 1, 0)$	7
Cheap	No	Yes	$(1, 0, 1)$	4
Expensive	Yes	Yes	$(0, 1, 1)$	9
Expensive	No	No	$(0, 0, 0)$	0
Expensive	No	Yes	$(0, 0, 1)$	3
Expensive	Yes	No	$(0, 1, 0)$	6

How Will I Make my Decision?

Let's say I don't care much about price, but I do care about my availability and interest. In this case, the corresponding weights might be $\bar{w} = (1, 6, 3)$. We can then compute the dot product $\bar{w} \cdot \bar{x}$ for each possible input:

Ticket Prices	Availability	Interest	$\bar{x}$	$\bar{w} \cdot \bar{x}$
Cheap	Yes	Yes	$(1, 1, 1)$	10
Cheap	No	No	$(1, 0, 0)$	1
Cheap	Yes	No	$(1, 1, 0)$	7
Cheap	No	Yes	$(1, 0, 1)$	4
Expensive	Yes	Yes	$(0, 1, 1)$	9
Expensive	No	No	$(0, 0, 0)$	0
Expensive	No	Yes	$(0, 0, 1)$	3
Expensive	Yes	No	$(0, 1, 0)$	6

We can now define my **activation threshold**, t, which will determine whether or not I go to the game, represented in binary.

Formula for Decision Making

The general formula for my decision to go to the Oilers game is

$$\text{output} = \begin{cases} 0 & \text{if } \bar{w} \cdot \bar{x} < t \\ 1 & \text{if } \bar{w} \cdot \bar{x} \geq t. \end{cases}$$

Formula for Decision Making

The general formula for my decision to go to the Oilers game is

$$\text{output} = \begin{cases} 0 & \text{if } \bar{w} \cdot \bar{x} < t \\ 1 & \text{if } \bar{w} \cdot \bar{x} \geq t. \end{cases}$$

For instance, if $t = 9$, we see I'll only go if I'm both available and interested.

Formula for Decision Making

The general formula for my decision to go to the Oilers game is

$$\text{output} = \begin{cases} 0 & \text{if } \bar{w} \cdot \bar{x} < t \\ 1 & \text{if } \bar{w} \cdot \bar{x} \geq t. \end{cases}$$

For instance, if $t = 9$, we see I'll only go if I'm both available and interested. If $t = 7$, I'll also go if the tickets are cheap and I'm available:

Ticket Prices	Availability	Interest	$\bar{x}$	$\bar{x} \cdot \bar{w}$
Cheap	**Yes**	**Yes**	$(1, 1, 1)$	**10**
Cheap	No	No	$(1, 0, 0)$	1
Cheap	**Yes**	**No**	$(1, 1, 0)$	**7**
Cheap	No	Yes	$(1, 0, 1)$	4
Expensive	**Yes**	**Yes**	$(0, 1, 1)$	**9**
Expensive	No	No	$(0, 0, 0)$	0
Expensive	No	Yes	$(0, 0, 1)$	3
Expensive	Yes	No	$(0, 1, 0)$	6

Perceptrons

This is a simplified model of a **perceptron**. The idea was developed by Frank Rosenblatt at Cornell in 1957, and is often used in psychology.

Perceptrons

This is a simplified model of a **perceptron**. The idea was developed by Frank Rosenblatt at Cornell in 1957, and is often used in psychology.

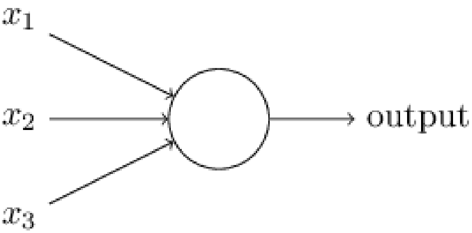

Figure 1: *Source: Nielsen*

Perceptrons

This is a simplified model of a **perceptron**. The idea was developed by Frank Rosenblatt at Cornell in 1957, and is often used in psychology.

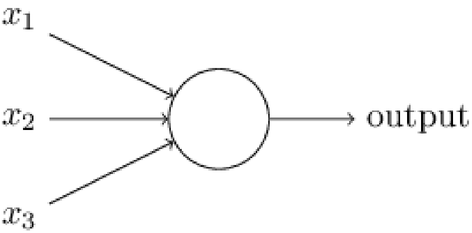

Figure 1: *Source: Nielsen*

Each of these lines collect evidence and are weighted to produce an output.

This is a simplified model of a **perceptron**. The idea was developed by Frank Rosenblatt at Cornell in 1957, and is often used in psychology.

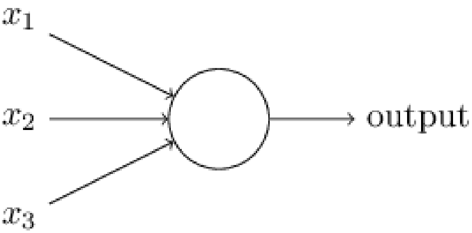

Figure 1: *Source: Nielsen*

Each of these lines collect evidence and are weighted to produce an output. In practice, our inputs and outputs don't necessarily have to be binary; they can be real-valued. We therefore have to define a new activation function.

Perceptrons

This is a simplified model of a **perceptron**. The idea was developed by Frank Rosenblatt at Cornell in 1957, and is often used in psychology.

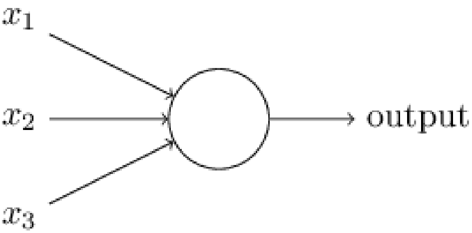

Figure 1: *Source: Nielsen*

Each of these lines collect evidence and are weighted to produce an output. In practice, our inputs and outputs don't necessarily have to be binary; they can be real-valued. We therefore have to define a new activation function. First, we make a slight modification to our model by adding bias.

Introducing the Bias

Instead of comparing our weighted sum to a threshold, we instead *add* a bias, b, to our weighted sum.

Introducing the Bias

Instead of comparing our weighted sum to a threshold, we instead *add* a bias, b, to our weighted sum. We write this as $\bar{w} \cdot \bar{x} + b$ instead.

Introducing the Bias

Instead of comparing our weighted sum to a threshold, we instead *add* a bias, b, to our weighted sum. We write this as $\bar{w} \cdot \bar{x} + b$ instead. Then

$$\text{output} = \begin{cases} 0 & \text{if } \bar{w} \cdot \bar{x} + b < 0 \\ 1 & \text{if } \bar{w} \cdot \bar{x} + b \geq 0. \end{cases}$$

Introducing the Bias

Instead of comparing our weighted sum to a threshold, we instead *add* a bias, b, to our weighted sum. We write this as $\bar{w} \cdot \bar{x} + b$ instead. Then

$$\text{output} = \begin{cases} 0 & \text{if } \bar{w} \cdot \bar{x} + b < 0 \\ 1 & \text{if } \bar{w} \cdot \bar{x} + b \geq 0. \end{cases}$$

Sometimes we'll need to calculate a value as an intermediate step in a calculation. In this case, we use various **activation functions**.

Introducing the Bias

Instead of comparing our weighted sum to a threshold, we instead *add* a bias, b, to our weighted sum. We write this as $\bar{w} \cdot \bar{x} + b$ instead. Then

$$\text{output} = \begin{cases} 0 & \text{if } \bar{w} \cdot \bar{x} + b < 0 \\ 1 & \text{if } \bar{w} \cdot \bar{x} + b \geq 0. \end{cases}$$

Sometimes we'll need to calculate a value as an intermediate step in a calculation. In this case, we use various **activation functions**. The one presented above is known as the *heaviside step function*. There's also the **rectified linear unit**, which is defined by $f(x) = \max\{0, x\}$.

Introducing the Bias

Instead of comparing our weighted sum to a threshold, we instead *add* a bias, b, to our weighted sum. We write this as $\bar{w} \cdot \bar{x} + b$ instead. Then

$$\text{output} = \begin{cases} 0 & \text{if } \bar{w} \cdot \bar{x} + b < 0 \\ 1 & \text{if } \bar{w} \cdot \bar{x} + b \geq 0. \end{cases}$$

Sometimes we'll need to calculate a value as an intermediate step in a calculation. In this case, we use various **activation functions**. The one presented above is known as the *heaviside step function*. There's also the **rectified linear unit**, which is defined by $f(x) = \max\{0, x\}$. Graphically,

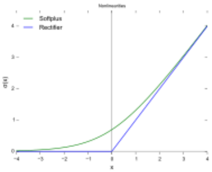

Figure 2: Rectifier, and a smooth approximation $\log(1 + e^x)$. (*Source: Wikipedia*).

Sigmoid Function

As we saw above, our output doesn't necessarily have to be a 0 or 1; using a rectified linear unit, it can be any non-negative number.

Sigmoid Function

As we saw above, our output doesn't necessarily have to be a 0 or 1; using a rectified linear unit, it can be any non-negative number. However, for computational purposes, it's easiest if our outputs live in the range $(0, 1)$.

Sigmoid Function

As we saw above, our output doesn't necessarily have to be a 0 or 1; using a rectified linear unit, it can be any non-negative number. However, for computational purposes, it's easiest if our outputs live in the range $(0, 1)$.

Outline

1 Decision Making

2 Classifying Digits through MNIST
- Defining the Problem
- References

Example Images

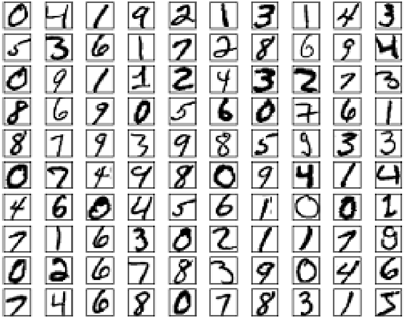

Figure 3: How would you devise a system for a **computer** to classify the digits? What assumptions do we have to make about the data set, known as MNIST?

Assumptions

- The MNIST database contains thousands of handwritten digits.

Assumptions

- The MNIST database contains thousands of handwritten digits.
- Each data-point contains both an image, and the desired digit.

Assumptions

- The MNIST database contains thousands of handwritten digits.
- Each data-point contains both an image, and the desired digit.
- Each image contains pixels ranging 0 to 255, with 8 bits used.
- An individual image is a 28×28 array of pixels.

Assumptions

- The MNIST database contains thousands of handwritten digits.
- Each data-point contains both an image, and the desired digit.
- Each image contains pixels ranging 0 to 255, with 8 bits used.
- An individual image is a 28×28 array of pixels.
- The desired digit is represented as a number from 0 to 9.

Assumptions

- The MNIST database contains thousands of handwritten digits.
- Each data-point contains both an image, and the desired digit.
- Each image contains pixels ranging 0 to 255, with 8 bits used.
- An individual image is a 28×28 array of pixels.
- The desired digit is represented as a number from 0 to 9.
- $60,000$ images are designated for training, and $10,000$ for testing.

Assumptions

- The MNIST database contains thousands of handwritten digits.
- Each data-point contains both an image, and the desired digit.
- Each image contains pixels ranging 0 to 255, with 8 bits used.
- An individual image is a 28×28 array of pixels.
- The desired digit is represented as a number from 0 to 9.
- $60,000$ images are designated for training, and $10,000$ for testing.

We'll build a model from the training images that will learn to classify digits!

What we're building towards

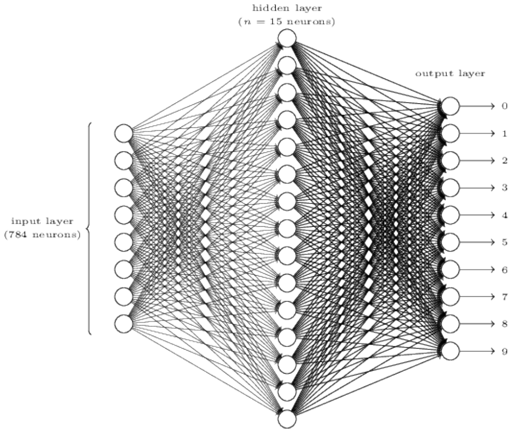

Figure 4: A simple neural network structure. The input vectors on the left hand side have $28 \times 28 = 784$ inputs for each pixel, and the output layer has 10 digits.

References

- Michael Nielsen: Using neural nets to recognize handwritten digits
- Towards Data Science: A Beginner's Guide to Neural Networks
- Paul's Online Math Notes, Calculus III