AI Bootcamp

RBMs and Recommendation Engines

Module 18 Day 3

Class Objectives

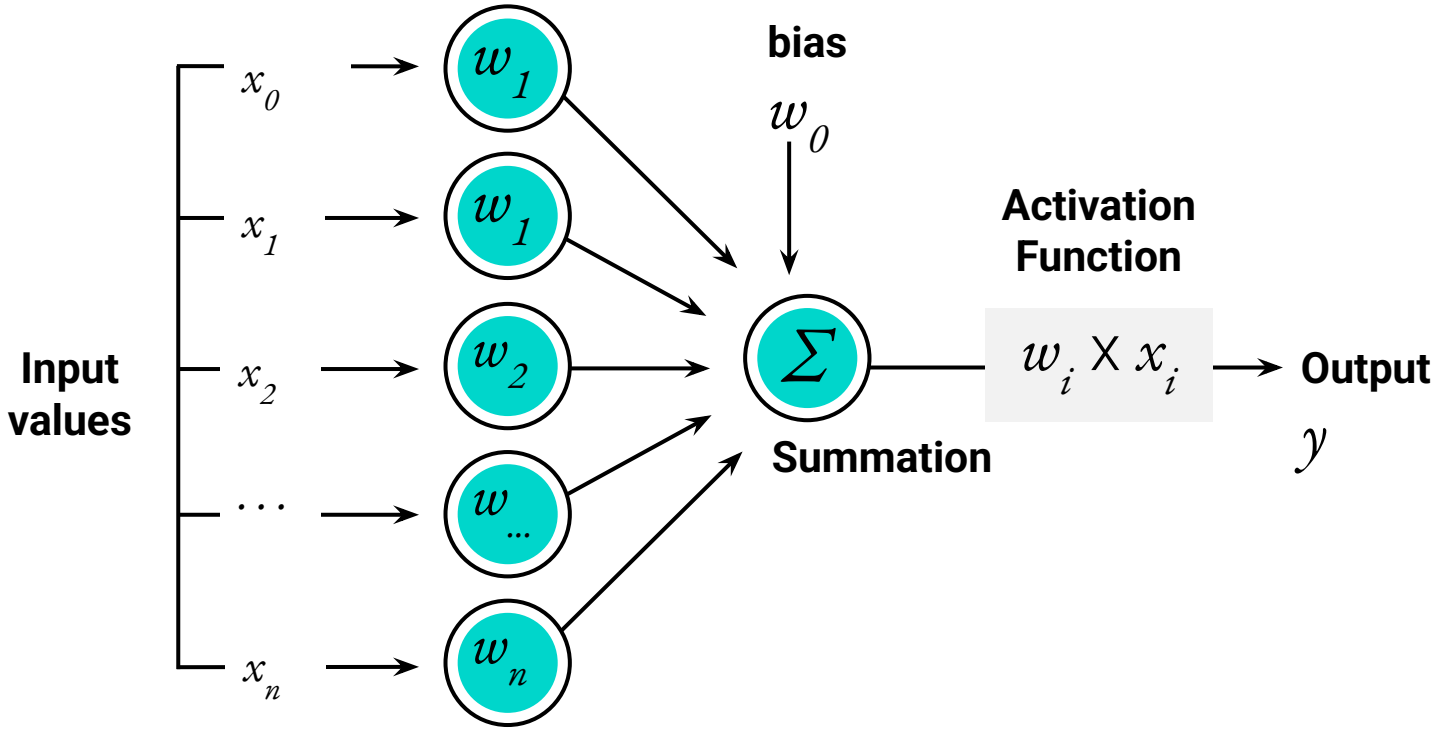
By the end of class, you will be able to:

1. Explain recommendation systems and their various implementations and challenges.
2. Use a Restricted Boltzmann Machine to develop a recommendation system.

Welcome

The Perceptron

The **perceptron model** mimics a biological neuron by receiving input data, weighting the information, and producing a clear output.



Instructor **Demonstration**

Introduction to Recommendation Systems

What are some examples of **recommendation systems** you interact with on a regular basis?

Recommendation Systems

Video recommendations

YouTube, Netflix,
Hulu, Disney+

Music recommendations

Spotify, YouTube Music

Shopping recommendations

Amazon, Wayfair,
your favorite online stores

Targeted ads

Suggested news articles

The platforms recommend what they think you might be interested in based on some previous behaviors. These recommendations are powered by **recommendation systems**.

Recommendation Systems

Neural network and deep learning models that can be used to develop recommendation systems, either as a component of the recommendation system, or the whole recommendation system model, include:

1. Multilayer perceptrons (MLPs)
2. Recurrent neural networks (RNNs)
3. Restricted Boltzmann Machines (RBMs)
4. Meta’s Deep Learning Recommender Model (DLRM)
5. Google's Wide & Deep
6. Neural Collaborative Filtering (NCF)
7. AutoEncoders

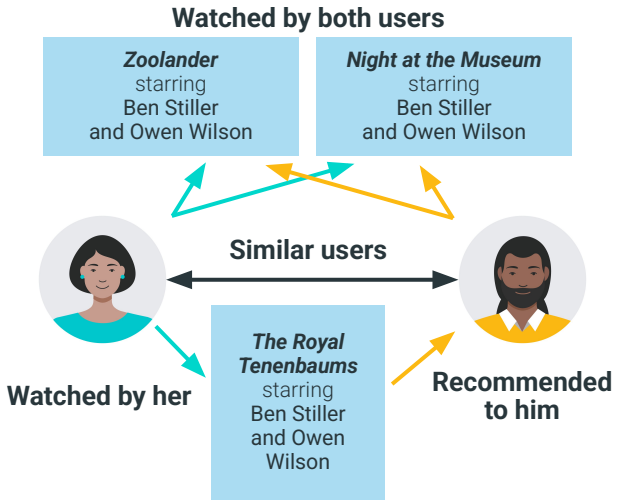
Recommendation systems are trained using data about **user behavior**, **preferences**, **interactions**, **and more**.

Types of Recommendation Systems

The first type of recommendation system we will consider is **collaborative filtering**.

01

In this model, the platform provides you with choices because a user that it deems to be similar to you has chosen that option before. The similarity is between users.



Source

02

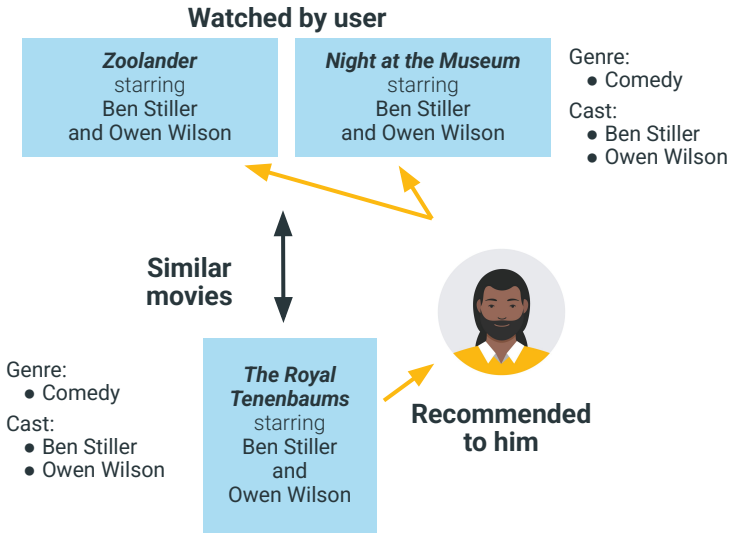
For example, if you are on a social networking platform such as Facebook or Instagram and you see a section recommending friends you should add to your network or profiles you could follow, this is likely because you have many mutual friends.

Types of Recommendation Systems

The second type of recommendation system we will consider is **content-based filtering**.

01

In this case, a platform will recommend something to you based on its similarity to items you've already indicated an interest in.



Source

02

Consider, for example, when you open a product page on an ecommerce site and you are shown a section titled "You may also like" or "Customers have also bought".

Types of Recommendation Systems

The last type of recommendation system is **context-based filtering**.

01

In such a recommendation system, the sequence of user actions combined with their context is used to inform recommendations.

02

For example, if a user watches *Stranger Things* on their laptop, *Master of None* on their tablet, and *Star Wars Rogue One* on their tablet, all in Australia, the model will try to predict the action the user will take next in the sequence.

Instructor **Demonstration**

RBM Basics

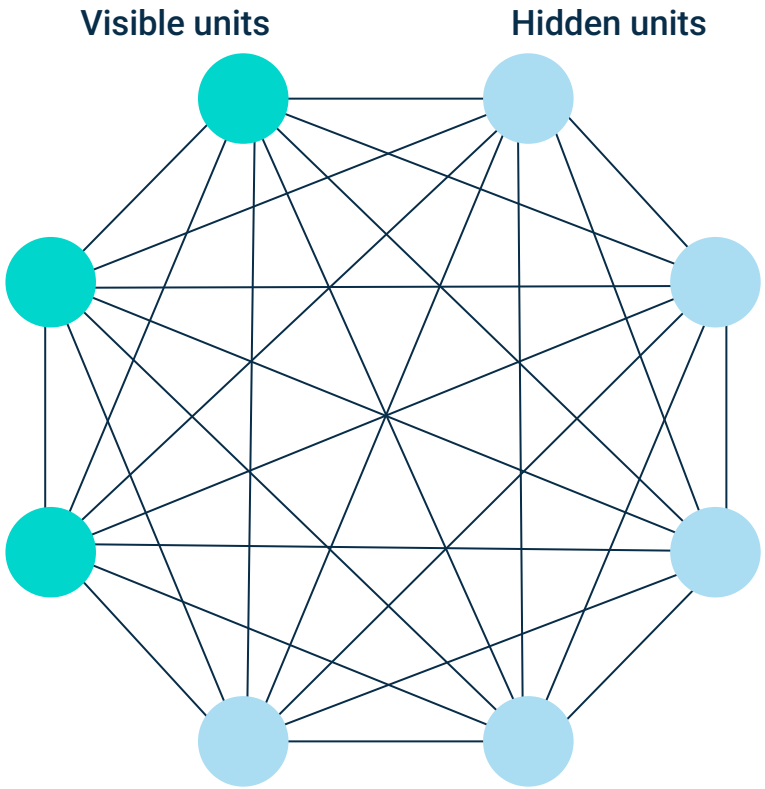
RBMs (Restricted Boltzmann Machine)

RBMs can be used in both supervised and unsupervised ways. Some applications of RBMs include:

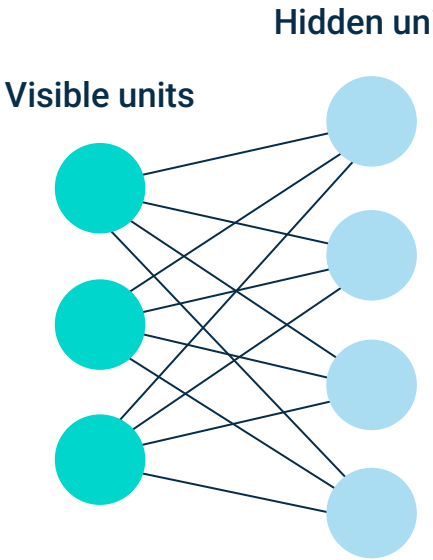
1. Dimensionality reduction
2. Classification
3. Collaborative filtering
4. Feature learning
5. Topic modeling

RBM Basics

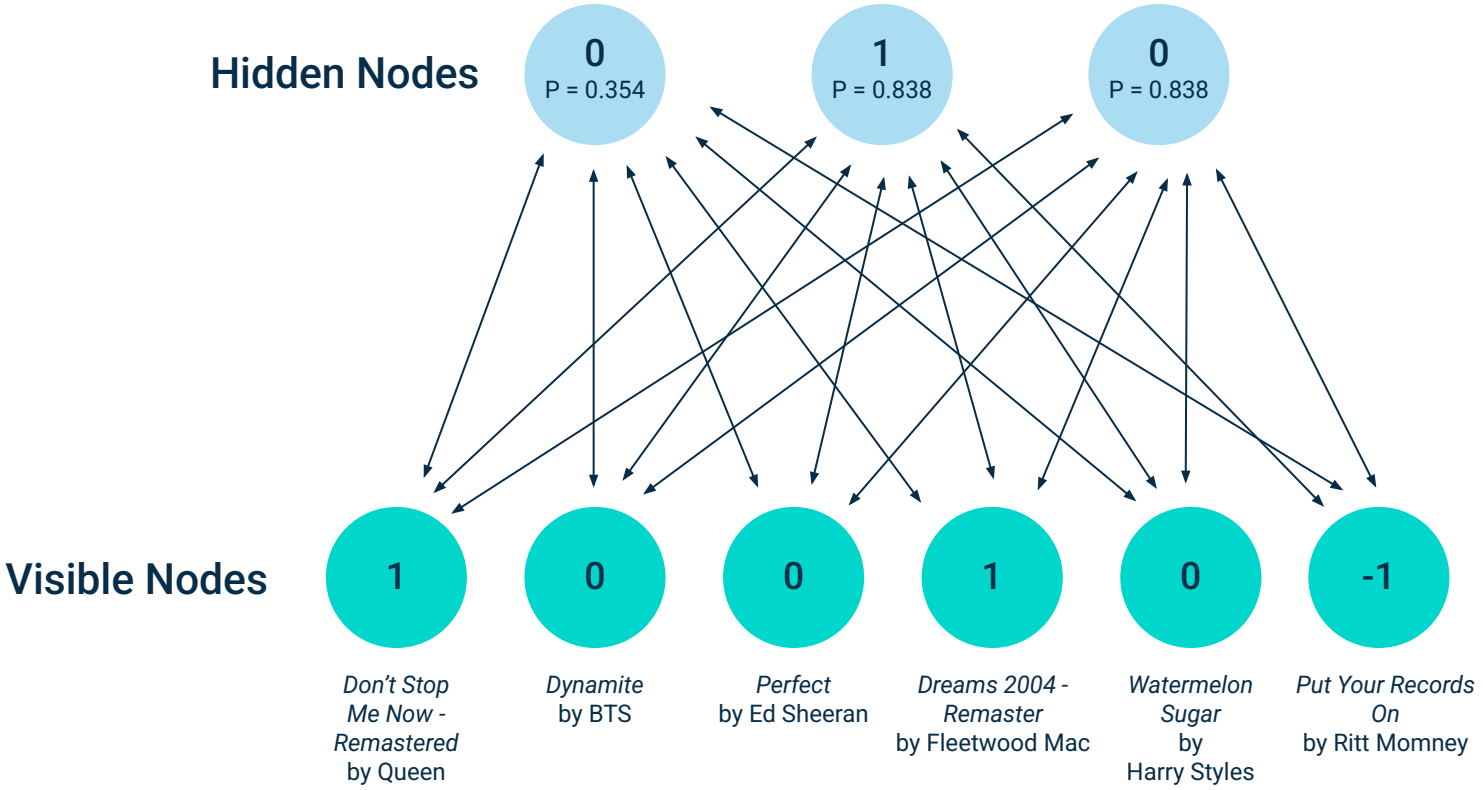
Boltzmann Machine



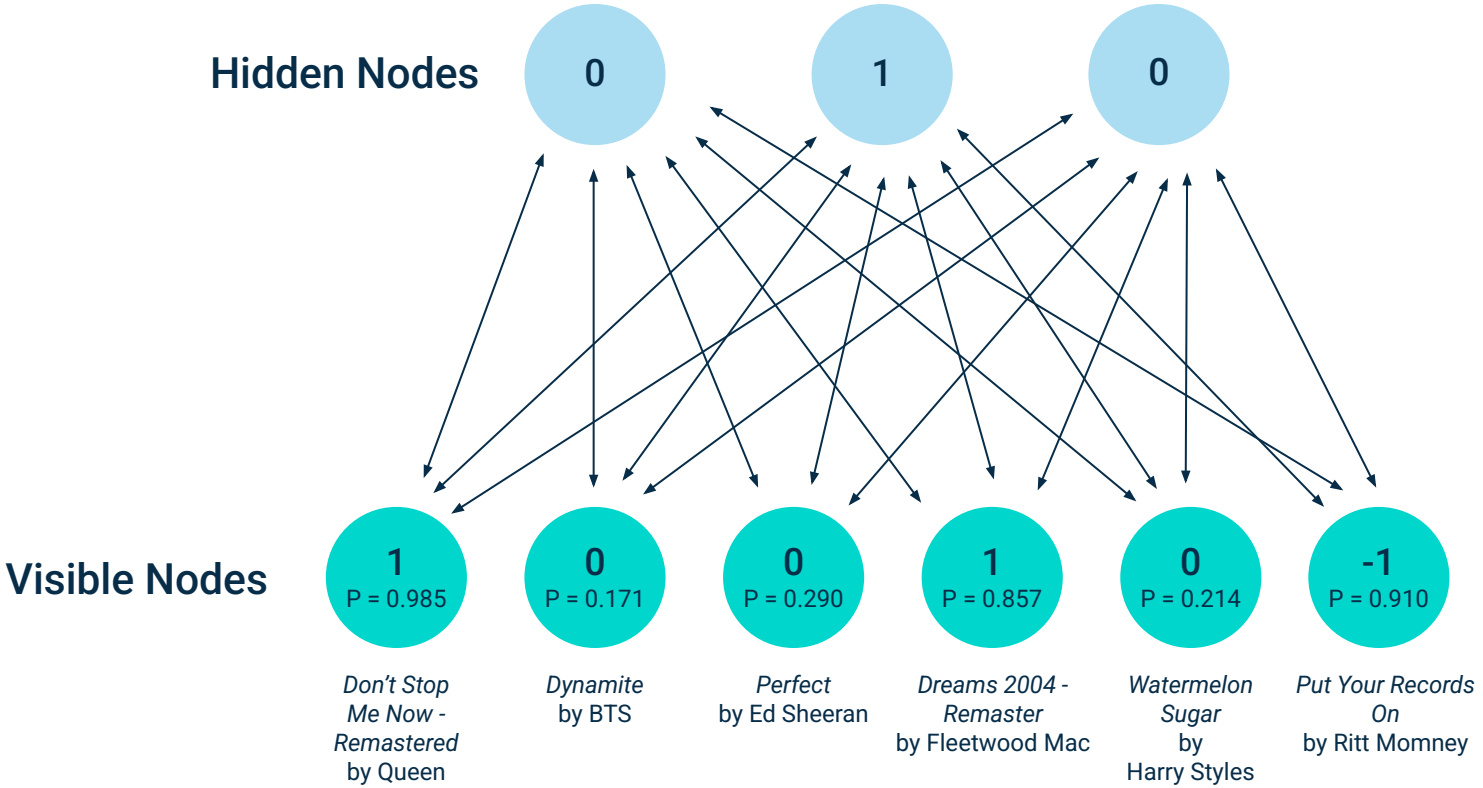
Restricted Boltzmann Machine



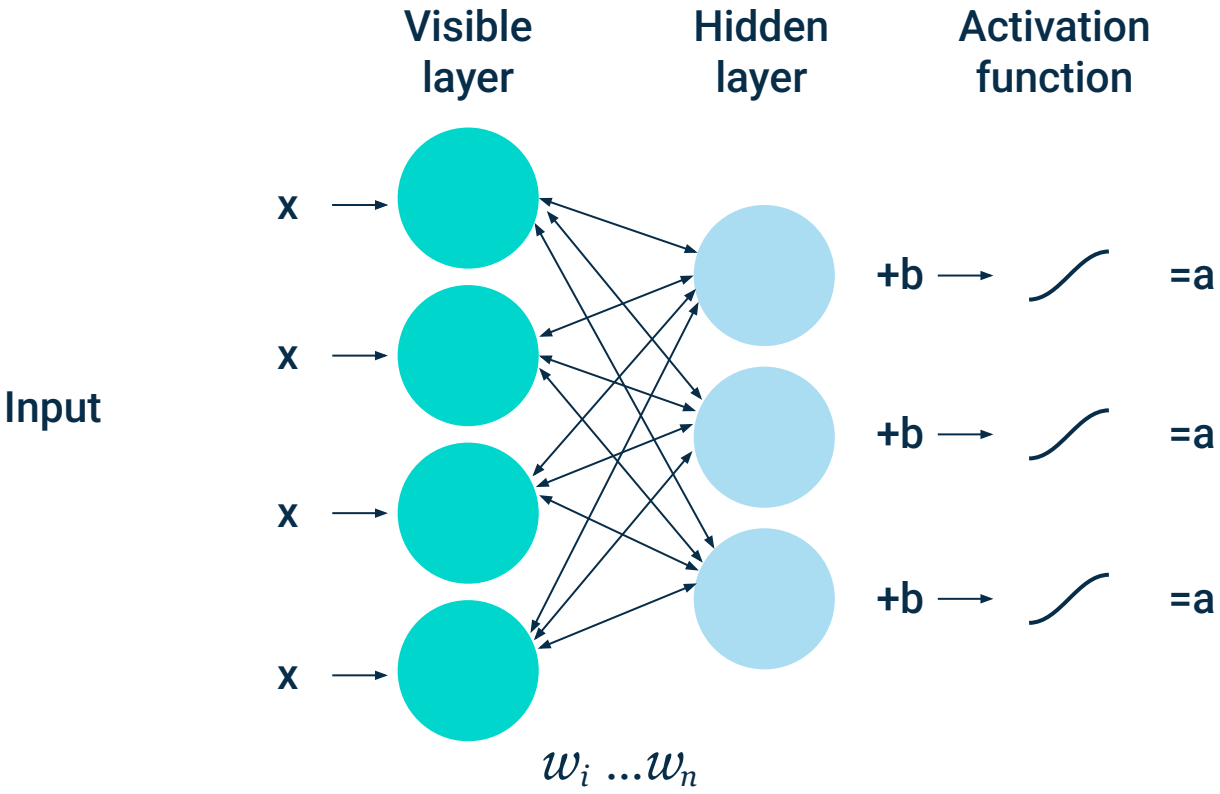
RBM Phase 1: Forward pass



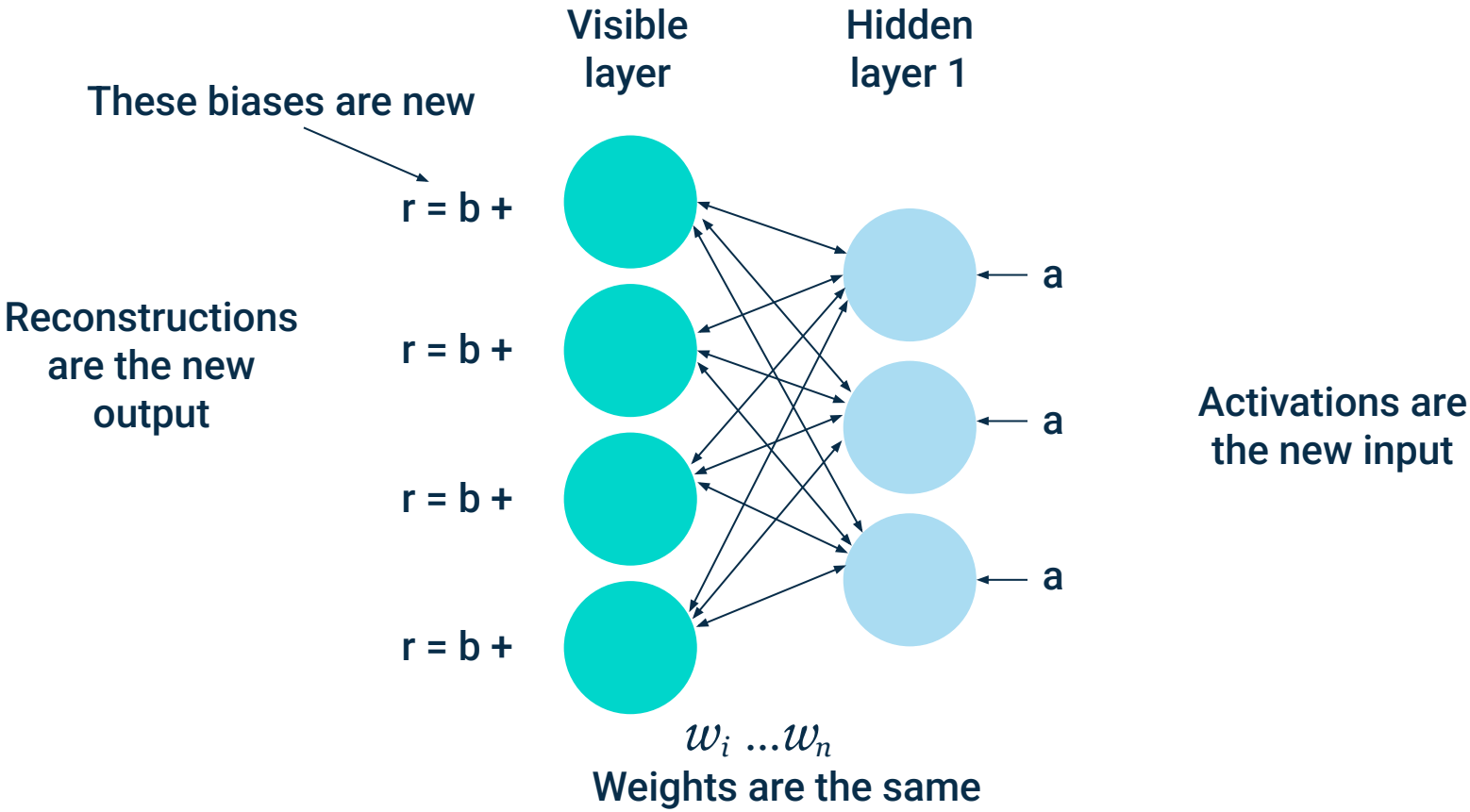
RBM Phase 2: Reconstruction



Input Processing: Forward Pass



Reconstruction: Back-propagation



Instructor **Demonstration**

Collaborative Filtering and Contrastive Divergence

Collaborative Filtering

User

Song

Song/User	Han Soon	Kya	Yindi	Suresh	Nuha	Jorge
Anti-Hero by Taylor Swift		4	4	3	4	3
Wonderwall by Oasis	3		2	5		5
Lullabies by Yuna		5		5	5	4
In Your Eyes by Kylie Minogue	1	5	5	4		
Let It Be by The Beatles	4	2	2	5		5
I Need U by BTS	5	4			4	

Collaborative Filtering

User

Song

Song/User	Han Soon	Kya	Yindi	Suresh	Nuha	Jorge
Anti-Hero by Taylor Swift		4	4	3	4	3
Wonderwall by Oasis	3		2	5		5
Lullabies by Yuna		5		5	5	4
In Your Eyes by Kylie Minogue	1	5	5	4		
Let It Be by The Beatles	4	2	2	5		5
I Need U by BTS	5	4			4	

Collaborative Filtering

User

Song

Song/User	Han Soon	Kya	Yindi	Suresh	Nuha	Jorge
Anti-Hero by Taylor Swift		4	4	3	4	3
Wonderwall by Oasis	3		2	5		5
Lullabies by Yuna		5		5	5	4
In Your Eyes by Kylie Minogue	1	5	5	4		
Let It Be by The Beatles	4	2	2	5		5
I Need U by BTS	5	4			4	

Collaborative Filtering

User

Song

Song/User	Han Soon	Kya	Yindi	Suresh	Nuha	Jorge
Anti-Hero by Taylor Swift		4	4	3	4	3
Wonderwall by Oasis	3	*2*	2	5	*2*	5
Lullabies by Yuna		5	*5*	5	5	4
In Your Eyes by Kylie Minogue	1	5	5	4	*5*	
Let It Be by The Beatles	4	2	2	5	*2*	5
I Need U by BTS	5	4	*4*		4	

Collaborative Filtering

User

Song

Song/User	Han Soon	Kya	Yindi	Suresh	Nuha	Jorge
Anti-Hero by Taylor Swift		4	4	3	4	3
Wonderwall by Oasis	3		2	5		5
Lullabies by Yuna		5		5	5	4
In Your Eyes by Kylie Minogue	1	5	5	4		*4*
Let It Be by The Beatles	4	2	2	5		5
I Need U by BTS	5	4			4	

User Preferences

Determining a user's preferences and saving that information for the model can be done in two ways:

01

Explicitly, such as when a user interacts with the system and gives an item a rating, likes a product, downvotes a post, or uses an emoji reaction on a Facebook post. The ratings we explored in the music example demonstrate explicit preferences.

02

Implicitly, such as when a user watches a video to the end, clicks on an ad, or spends a few seconds reading a post rather than scrolling right past it. All of these examples could indicate that the user is interested in more videos, ads, or posts like the ones they gave attention to.

Training with Contrastive Divergence

W	Weights
v_0	Visible inputs
h_0	Hidden states after the forward pass
v_1	Reconstructed inputs
h_1	Resampled hidden states from the reconstructed input
α	Learning rate

Updating the weights:

$$\Delta W = v_0^T h_0 - v_1^T h_1$$

$$W = W + \alpha \times \Delta W$$

Updating the biases:

$$\text{bias}_{vis} = \text{bias}_{vis} + \alpha \times (v_0 - v_1)$$

$$\text{bias}_{hid} = \text{bias}_{hid} + \alpha \times (h_0 - h_1)$$

Instructor **Demonstration**

Building a Model in TensorFlow without Keras

Tensors

Tensors make it possible to create vector and matrix representations in our programs. Tensors allow us to store arrays in multiple dimensions, and we will make use of tensors to store weights, biases, and input data.

To understand this, let's consider the list of values `[1.0, 2.0, 3.0]`. We could make this into a **vector** or a **rank-1 tensor** using `tf.constant()` as follows:

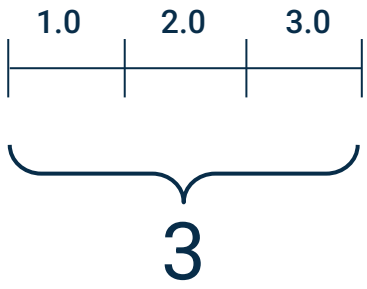
```
import TensorFlow as tf

# Create a rank-1 tensor
t_1 = tf.constant([1.0, 2.0, 3.0])
print(t_1)
```

Which prints the following output:

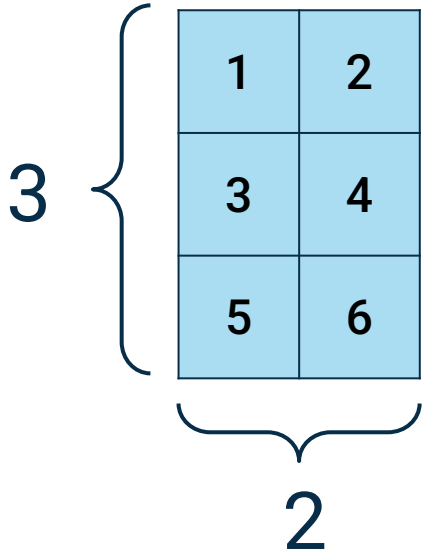
```
tf.Tensor([1. 2. 3.], shape=(3,), dtype=float32)
```

Tensor Representations

Vector
1 dimension
shape(3,)

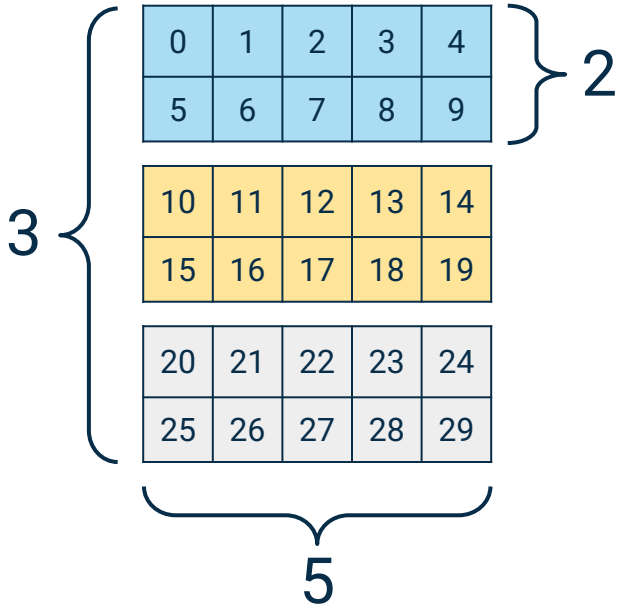


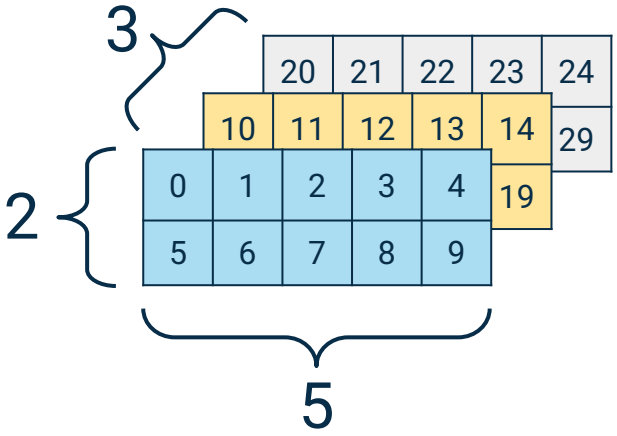
Matrix or array
2 dimensions
shape(3, 2)

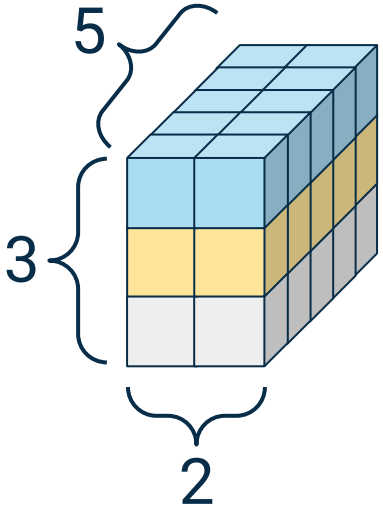


Tensor Representations *continued*

3-axis-tensor
3 dimensions
shape(3, 2, 5)







Creating Tensors

There are a few ways to create tensors in TensorFlow and as usual, the most appropriate option will depend on the context of the use-case:

`tf.convert_to_tensor()`	Can be used to convert other formats, such as Python lists, numpy arrays, or pandas DataFrames, into tensors.
`tf.constant()`	Creates an immutable tensor. Use this if you don't need to change or update the tensors in any way.
`tf.Variable()`	Creates a tensor that can be modified. Similar to Python variables, we can initialize a Variable tensor and then update with new values later on in the program. Variable tensors help us store and update our weights and bias tensors.

Tensor Representations

We use can use the following functions to perform relevant mathematical operations on our tensors:

`tf.matmul()`	Performs matrix multiplications such as when the inputs are multiplied by the model weights.
`tf.transpose()`	Transposes a matrix, i.e., changes the dimensions of the matrix. For example, a 2 x 1 matrix would be transposed to a 1 x 2 matrix. Recall the equation: $\Delta W = v_0^T h_0 - v_1^T h_1$ The superscript T stands for transpose. Sometimes we need to transpose a matrix or vector in order to perform a matrix multiplication.

Activation Functions in TensorFlow

Activation functions we previously used within the Keras module are also available in the TensorFlow **nn** module.

```
tf.nn.relu()

tf.nn.leaky_relu()

tf.nn.sigmoid()

tf.nn.tanh()
```

When training models with massive amounts of data, we often have to create slices of data called **batches**, and train the model with each of the batches.

Batches

To convert our training data into batches, we can use the following code:

```
tf.data.Dataset.from_tensor_slices((np.float32(X_train))).batch(batchsize)
```

`X_train`	This is our training data.
`np.float32()`	Converts the training data to a numpy array of data type float32.
`batchsize`	The parameter that specifies the number of rows of data to include in a batch.
`.from_tensor_slices()`	Creates a dataset whose elements are slices of the given tensors. The given tensors are sliced along their first dimension. This operation preserves the structure of the input tensors, removing the first dimension of each tensor and using it as the dataset dimension.
`.batch()`	Separates the tensor sliced dataset into batches according to the batchsize argument.

Source

Instructor **Demonstration**

RBM Training

Questions?

Instructor **Demonstration**

Preparing Data for RBMs

Sparse Ratings Matrix

Since it is unlikely that a user will have rated a significant number of items in the data catalogue, a ratings matrix will have many empty cells.

Movie

User

User/Movie	*Hero*	*Turning Red*	*Barbie: The Movie*	*Elemental*	*The Goonies*	*Selma*
Han Soon	5			4		
Kya		5				
Yindi			4			
Suresh				3	4	
Nuha		4			2	
Jorge						5

Storing Ratings Data

It is common to store only the ratings data that exists, but this means we must convert the stored data to a ratings matrix.

	User	Movie	Rating
1	Han Soon	*Hero*	5
2	Han Soon	*Elemental*	4
3	Kya	*Turning Red*	5
4	Yindi	*Barbie: The Movie*	4
5	Suresh	*Elemental*	3
6	Suresh	*The Goonies*	4
7	Nuha	*Turning Red*	4
8	Nuha	*The Goonies*	2
9	Jorge	*Selma*	5

Data Preprocessing

Once we have our ratings matrix, we must prepare it for training. This means filling in the null cells, and normalizing the ratings.

Movie

User

User/Movie	*Hero*	*Turning Red*	*Barbie: The Movie*	*Elemental*	*The Goonies*	*Selma*
Han Soon	1.0	0.0	0.0	0.8	0.0	0.0
Kya	0.0	1.0	0.0	0.0	0.0	0.0
Yindi	0.0	0.0	0.8	0.0	0.0	0.0
Suresh	0.0	0.0	0.0	0.6	0.8	0.0
Nuha	0.0	0.8	0.0	0.0	0.4	0.0
Jorge	0.0	0.0	0.0	0.0	0.0	1.0

Questions?

Break

15 mins

Instructor **Demonstration**

Training the Movie Recommender, Making Predictions, and Saving the Model

Questions?

Instructor **Demonstration**

Create Utilities to Reuse the Model

Questions?

Instructor **Demonstration**

Evaluation Metrics for Recommendation Systems

Questions?

Instructor **Demonstration**

Use the Model to Make Recommendations

Questions?

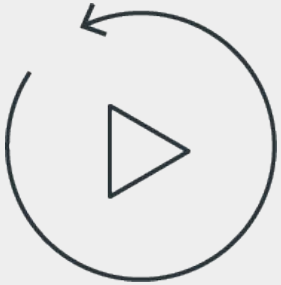
Instructor **Demonstration**

Real-World Considerations

Real World Challenges

In addition to the general trade offs to consider, recommendation systems have their own set of challenges.

1. Cold-start problem
2. Data sparseness
3. Frequency of model training and updates
4. Diversity of recommendations
5. Privacy and trust
6. Changing user interests
7. Scalability and robustness
8. Filter bubbles



Let’s **recap**

Recap

After today's lesson you are able to:

1. Explain recommendation systems and their various implementations and challenges.
2. Use a Restricted Boltzmann Machine to develop a recommendation system.

Questions?

The End