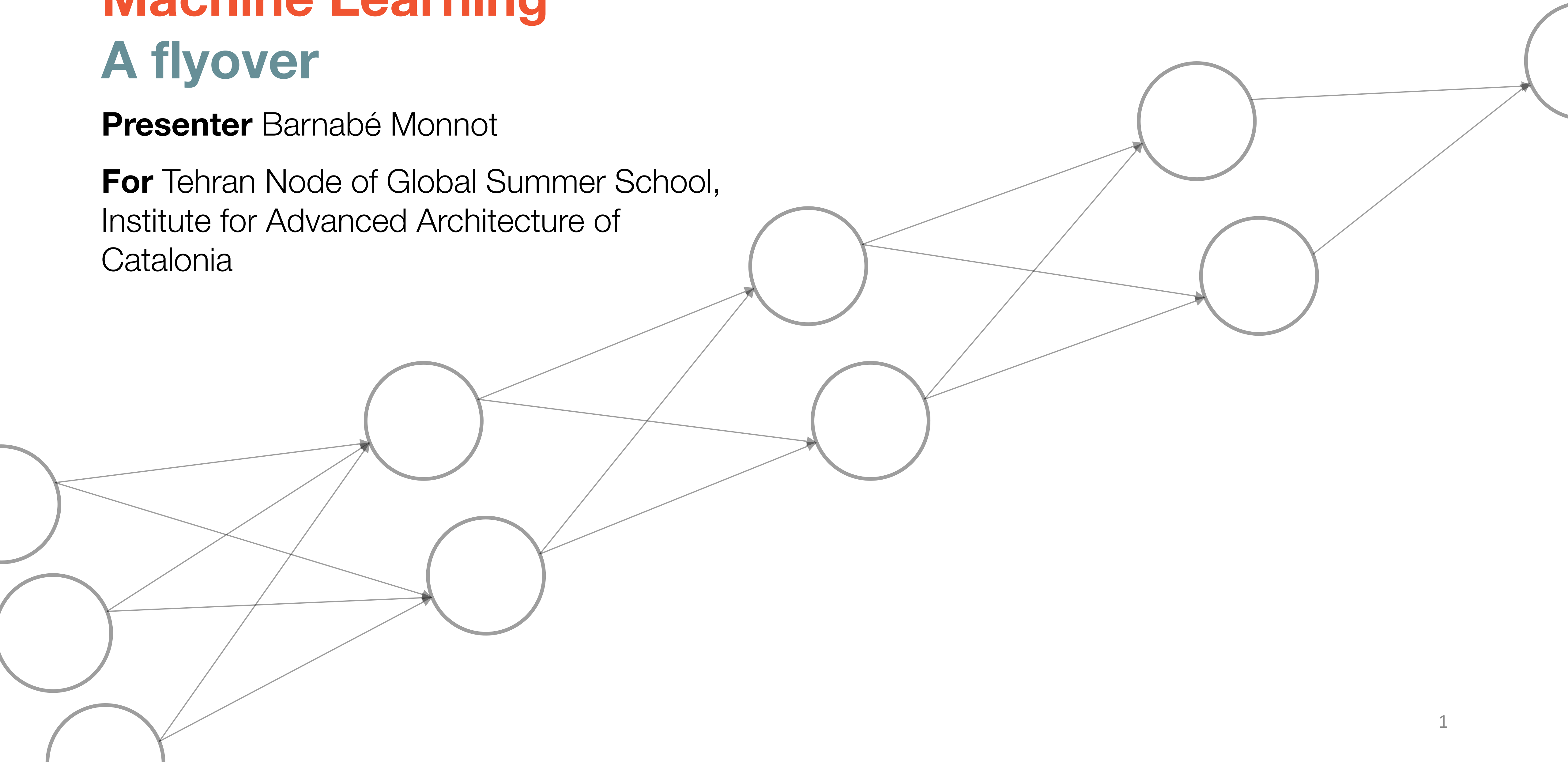
Machine Learning
A flyover

Presenter Barnabé Monnot

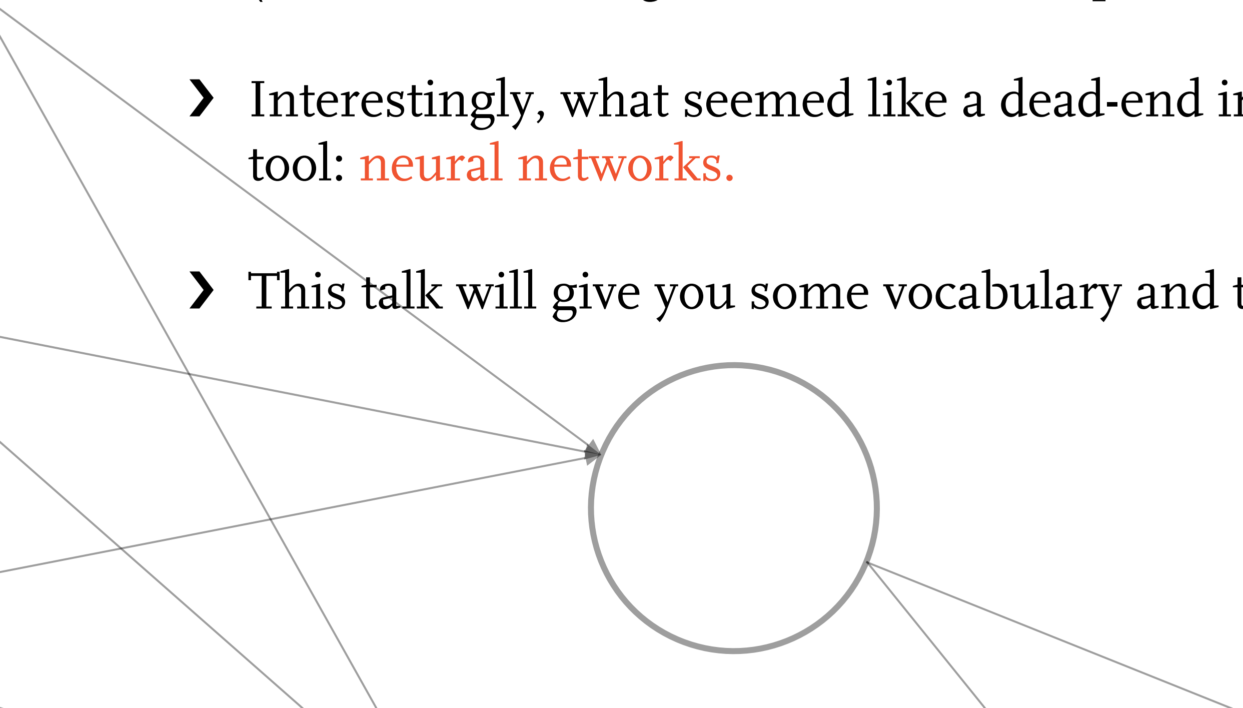
For Tehran Node of Global Summer School, Institute for Advanced Architecture of Catalonia



Machine Learning

Using data to extract patterns, learn new structures and make predictions

- Earliest results in Machine Learning as old as late 1950's (e.g. Perceptron)
- Since then, went through several mutations, some of them explained by the available tech (better GPUs, custom chips), some of them due to the current "hot" research topics (neuroscience, cognitive science, computer vision).
- Interestingly, what seemed like a dead-end in the 80's is now one of ML's most successful tool: neural networks.
- This talk will give you some vocabulary and tools to decode what practitioners talk about!



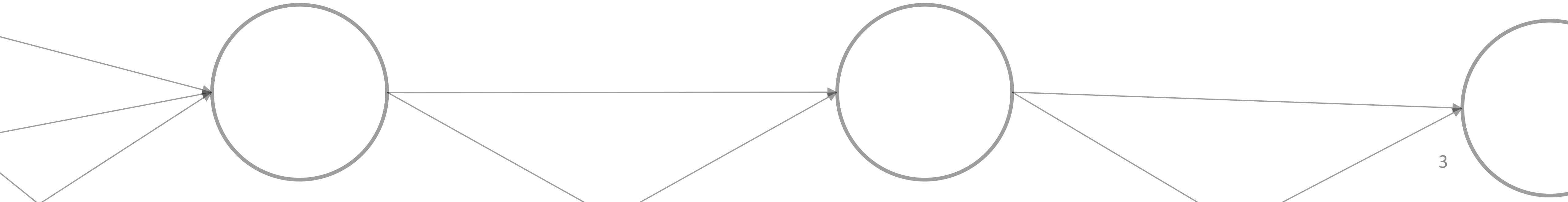
Contents of the talk

Machine Learning basics

- The Data zoo: Supervised / Unsupervised
- Modeling, training, predicting

The Model zoo

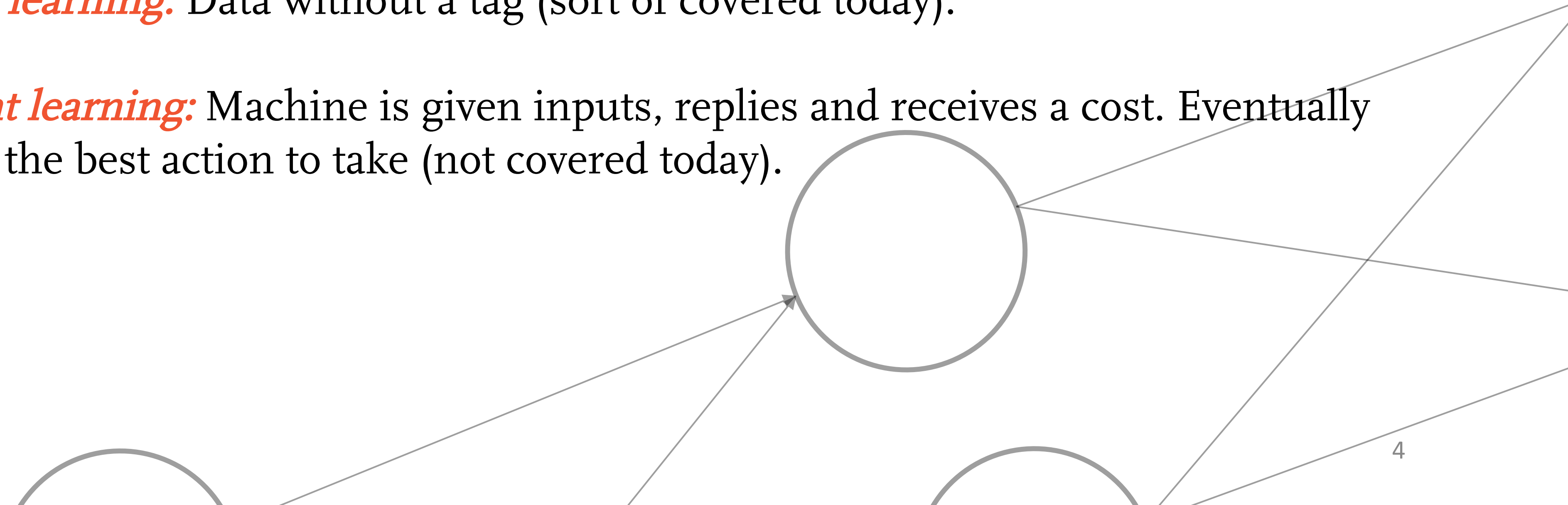
- Collaborative filtering: Predicting a user rating
- Neural networks and deep learning: Dog or cat?



The Data zoo

A machine takes inputs to produce outputs.

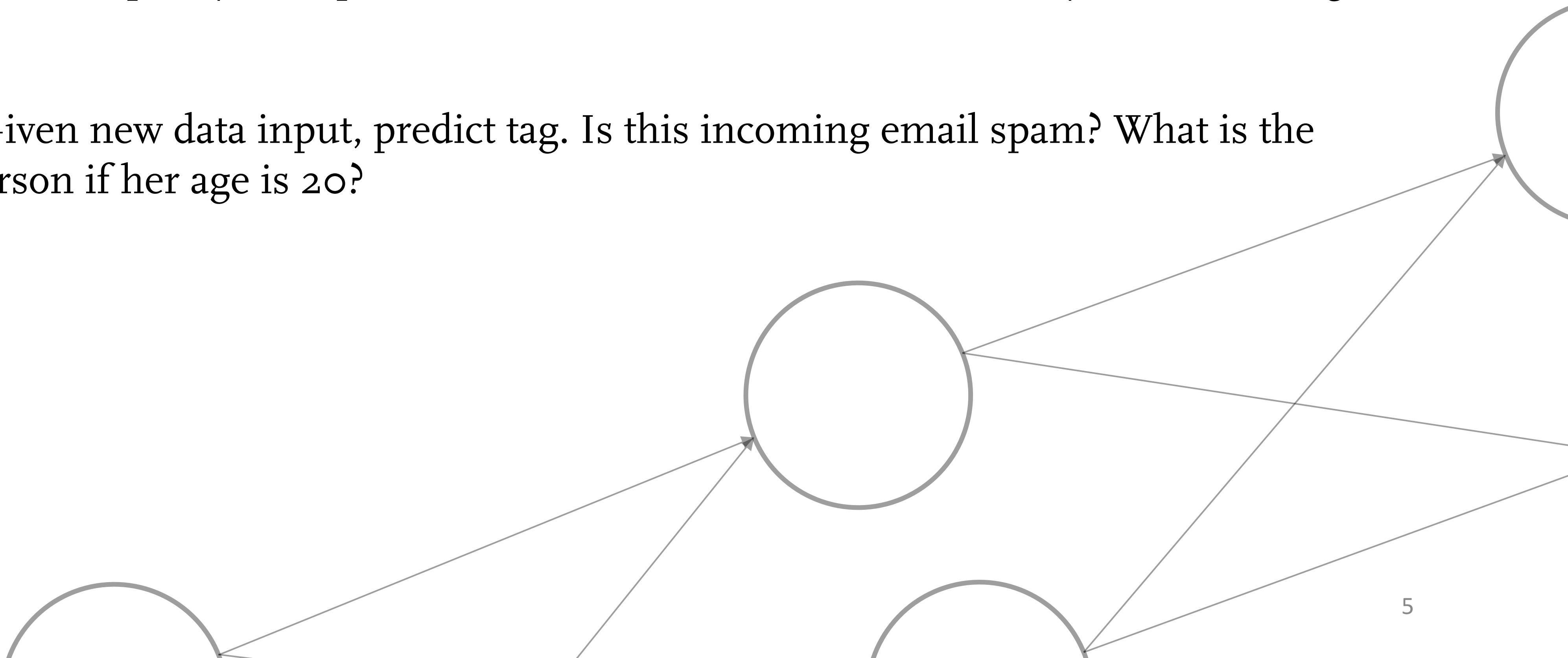
- Data comes in various forms, and each variation brings different challenges.
- We traditionally divide ML problems in three main categories:
  - *Supervised learning:* Data that has a tag. Input → Tag (covered today).
  - *Unsupervised learning:* Data without a tag (sort of covered today).
  - *Reinforcement learning:* Machine is given inputs, replies and receives a cost. Eventually learns what is the best action to take (not covered today).



Supervised learning

Data is labeled and we predict label of new inputs

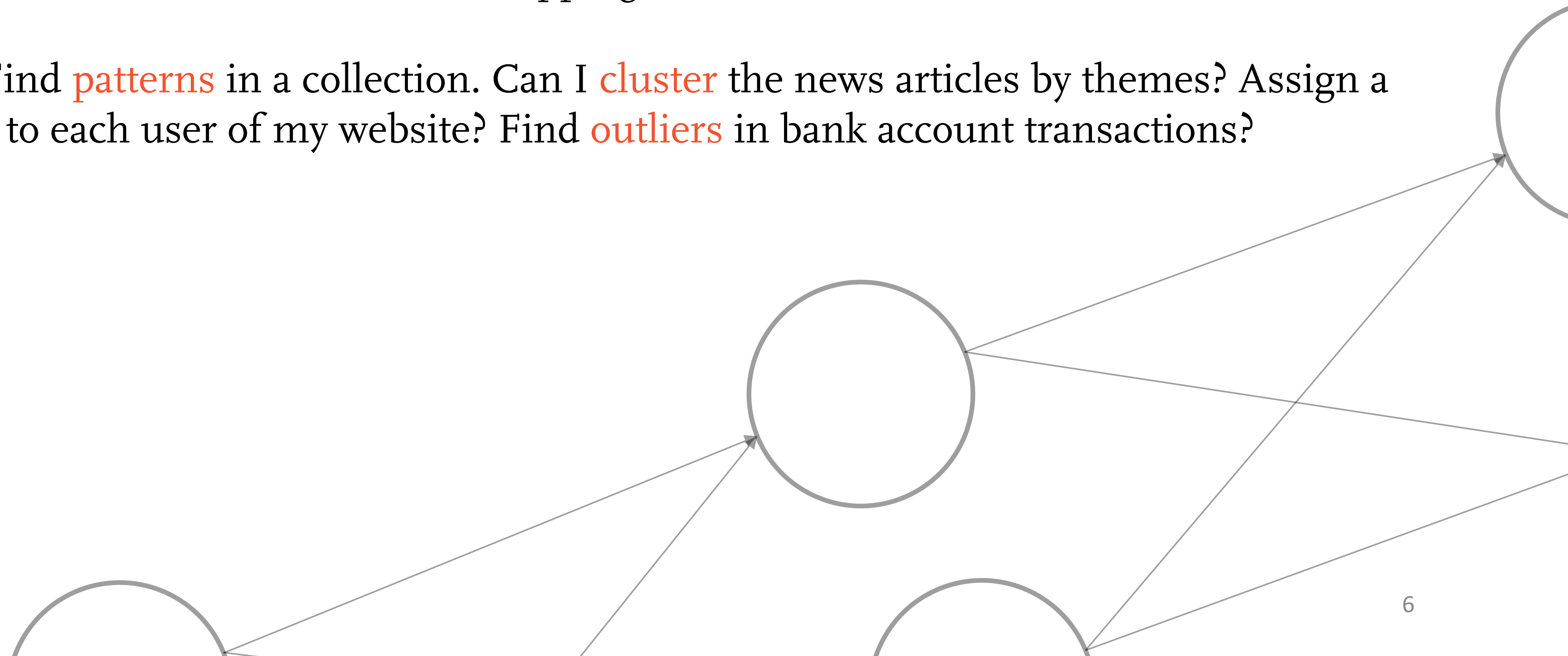
- *Supervised learning:* Data that has a tag. Input → Tag.
- Examples: Email → Spam / Not spam ; Bank account transaction → Fraud / Not fraud ; Age → Weight.
- Usual tasks: Given new data input, predict tag. Is this incoming email spam? What is the weight of a person if her age is 20?



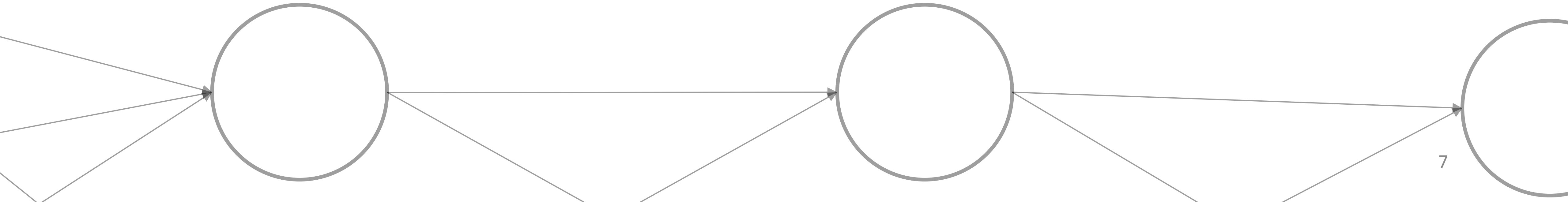
Unsupervised learning

Unlabeled data and we classify new inputs

- *Unsupervised learning:* Data without a tag.
- Examples: News articles ; User data on a shopping website ; Bank account transactions.
- Usual tasks: Find patterns in a collection. Can I cluster the news articles by themes? Assign a typical profile to each user of my website? Find outliers in bank account transactions?



Contents of the talk



Modeling (Supervised)

Model

ML teaches the machine to fit the best model to a collection of data

- We have an input *x*. We believe that there exists a (possibly very complex) relationship between the input *x* and its tag *y*. Call this relationship *f*.

$$f(x) = y$$

- In the previous examples, *x* is an email, *y* is Spam or Not spam
  x is a picture, *y* is Dog / Cat.

- We want to learn from the data the function *f*.

- ML simply gives different techniques to find this function *f*.

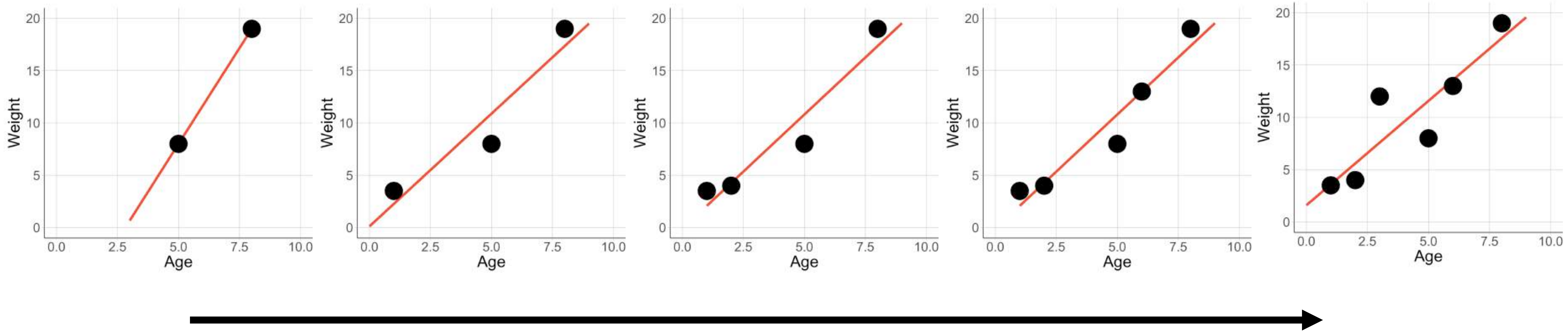
Training (Supervised)

Train

- I believe age and weight are related by the function *f*

$$weight = f(age) = b_o + age * b_I$$

- I.e, I am trying to find the line of best fit between age and weight (a.k.a do a regression).
- This is called training: Finding the best values of b_o and b_I to represent the data.

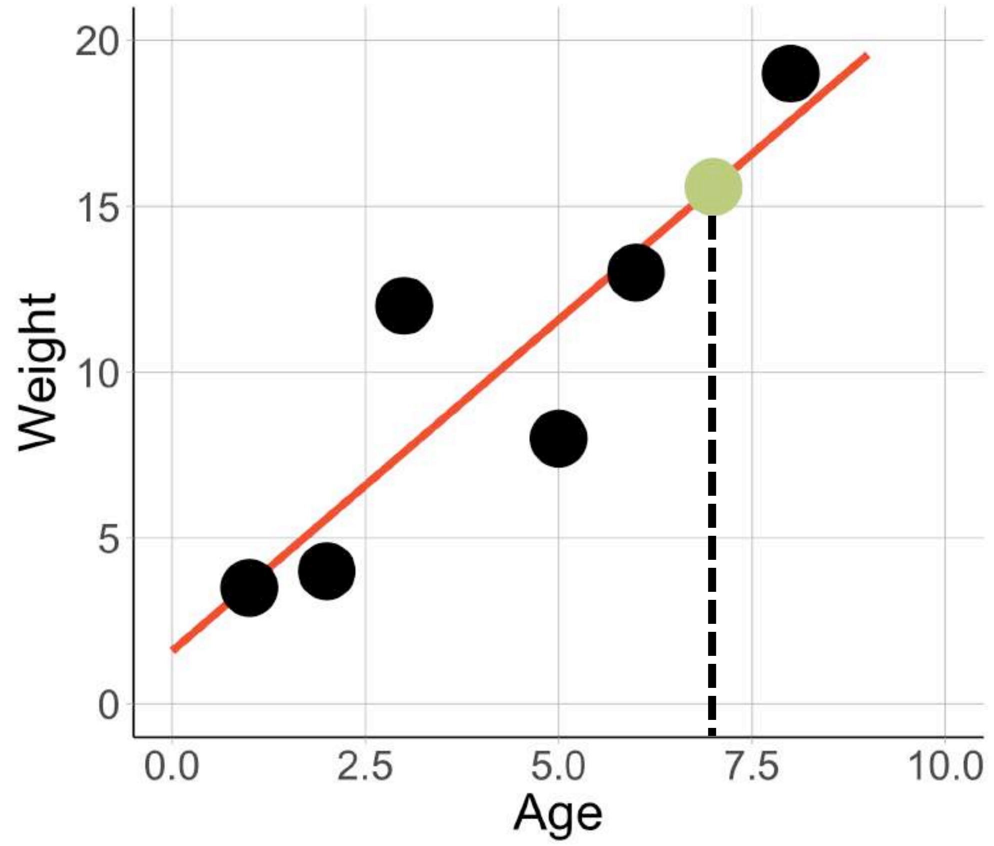


More data

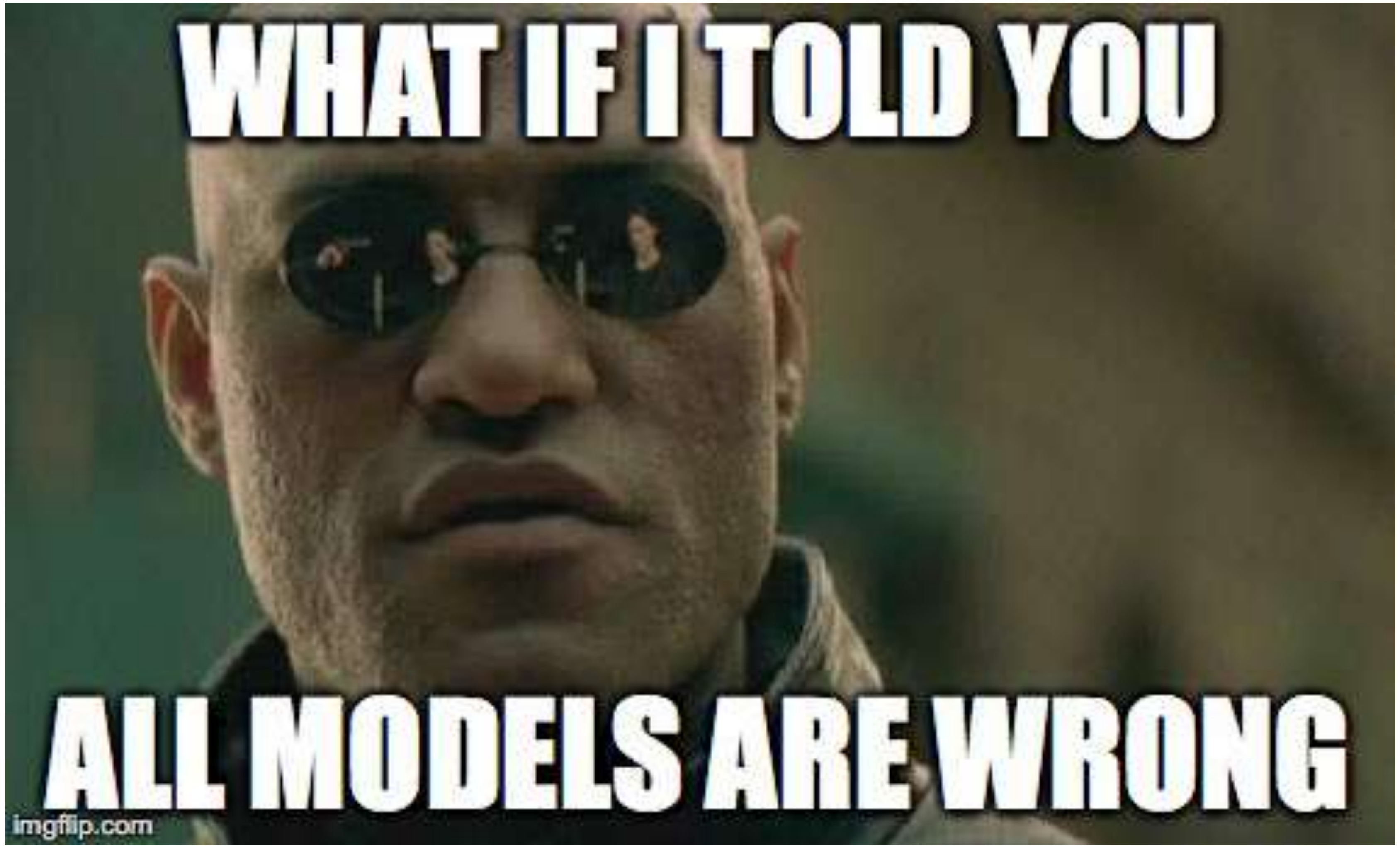
Predicting (Supervised)

Predict

- Once we have fitted the model to the data, we can make predictions!

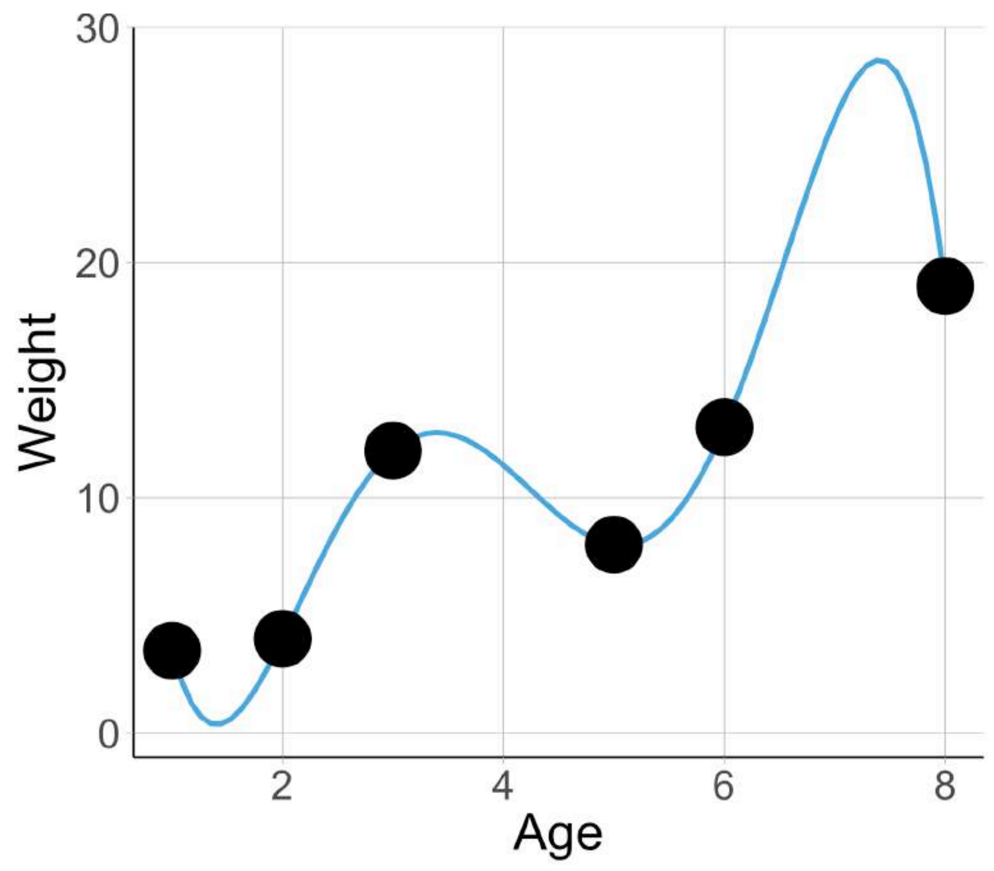


- We want to know the weight of a 7-year-old child according to our model.
- The green point shows our best prediction according to the model.
- Keep this figure in mind! Useful approximation of most ML algorithms.
  - We have data points.
  - We decide on the model.
  - We fit the model to the data (training).
  - We predict!


WHAT IF I TOLD YOU
ALL MODELS ARE WRONG
imgflip.com

Overfitting (Supervised)

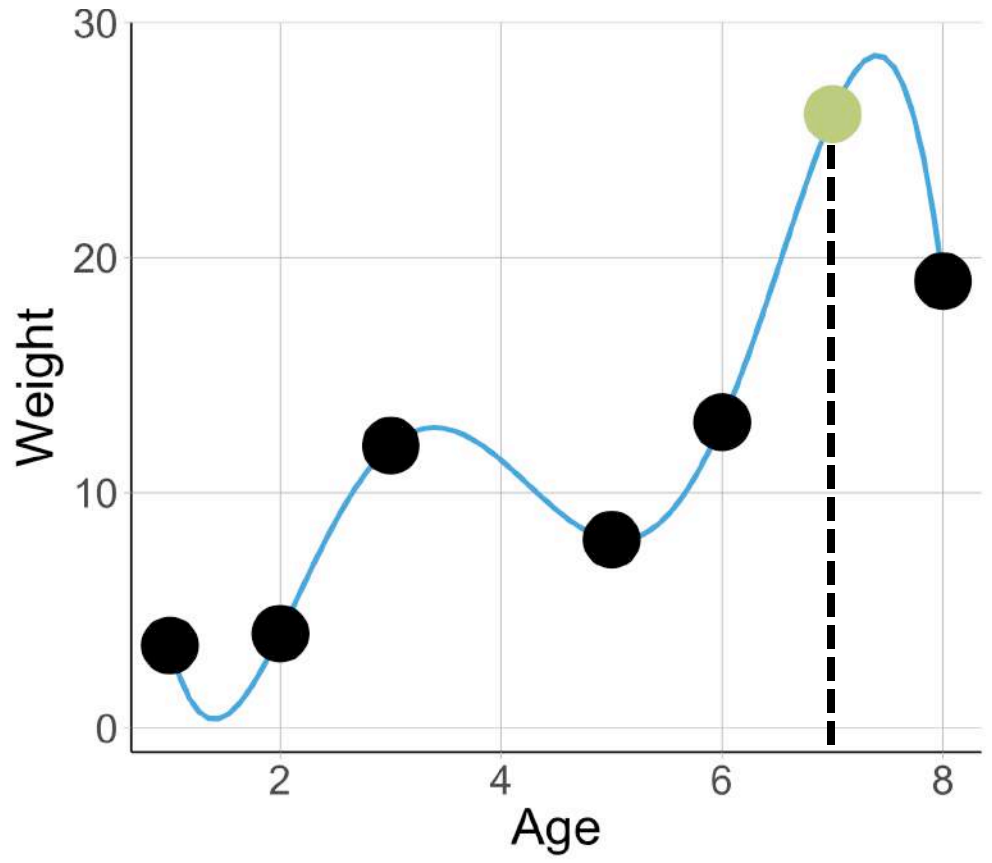
- But I can make a model that is not wrong!



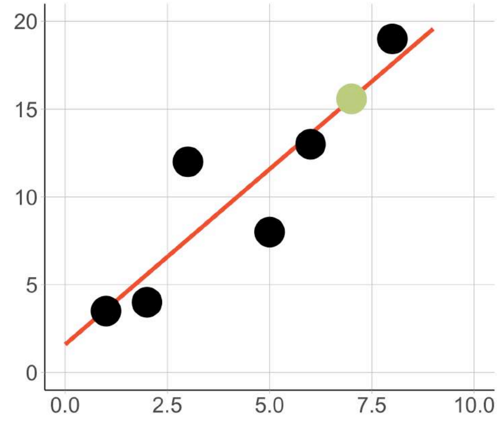
- This model is very good! Perfectly explains all of our data points, the curve goes through them.
- (Can be obtained by doing a polynomial interpolation for example)
- So what is wrong? Why do we not want such a model?

Overfitting (Supervised)

- It makes terrible predictions!



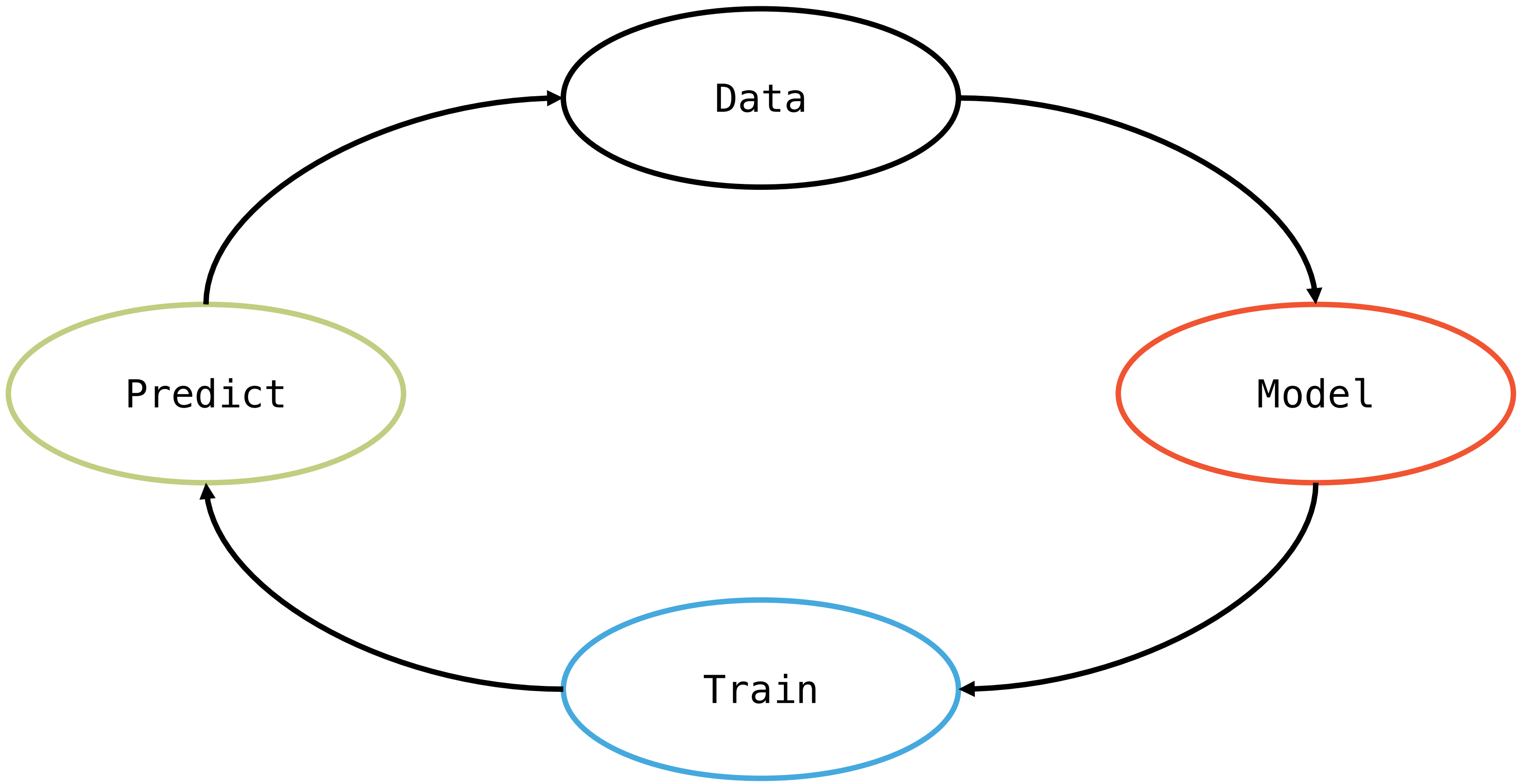
- Now with this model, we predict the weight of a 7 year old to be at the green dot.
- Compare with our previous model:



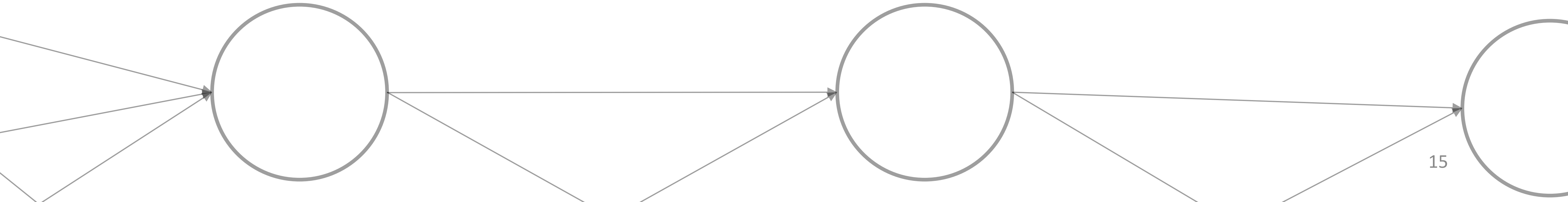
- This seems unrealistic doesn't it?
- A good Machine Learning algorithm will decide that the first model is better than the second one!

What we saw

ML is the process of using data to build predictive models

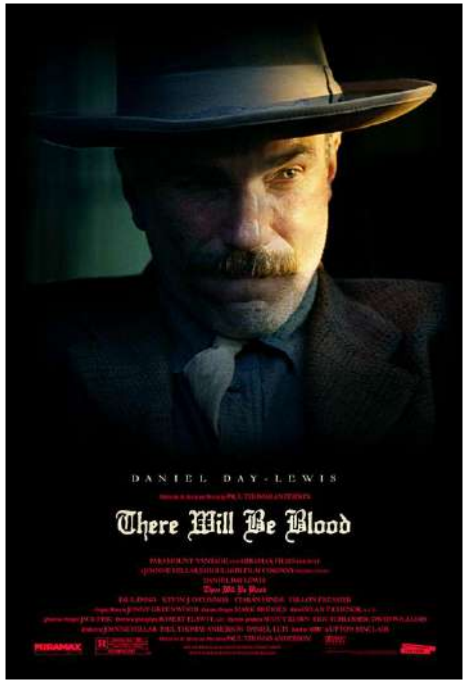
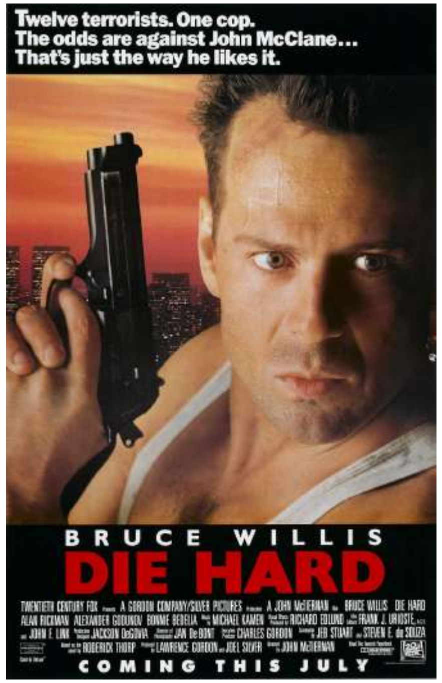


Contents of the talk



Collaborative filtering: Our dataset

› *N* users and *M* movies. Each user can give rating 1 to 5. Can we predict the missing ratings?

				
Alice	★★★★	★	???	★★★★★
Bob	???	★★★★★	★★★	★

DANIEL DAY-LEWIS
There Will Be Blood
Twelve terrorists. One cop. The odds are against John McClane... That's just the way he likes it.
BRUCE WILLIS
DIE HARD
COMING THIS JULY
BLADE RUNNER
WOODY ALLEN DIANE KEATON TONY ROBERTS CAROL KANE PAUL SIMON JANET MARGOLIN SHELLEY DUVALL CHRISTOPHER WALKEN COLLEEN DEWHURST
"ANNIE HALL"
A nervous romance.

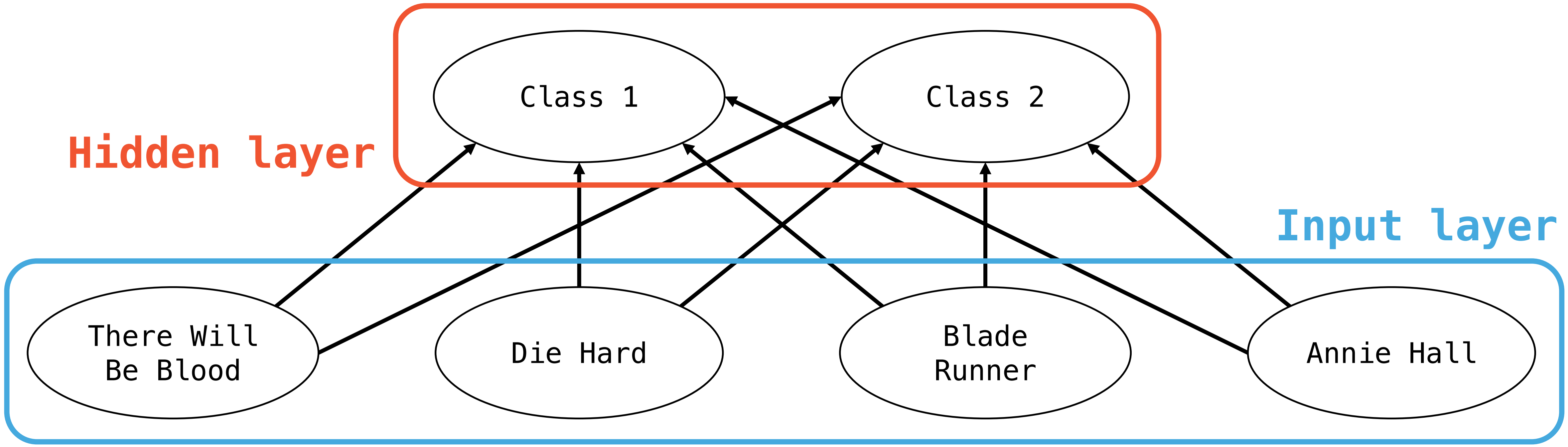
Collaborative filtering: Intuition

Find similar movies from the ratings

- The idea: find the common denominator between movies using the ratings only.
- Say that two movies are similar if similar users have rated them similarly. How many classes of movies do I need to discrimate between all of them?
- A very simple model: one class, all movies are in it → Underfitting!
  A very complex model: M classes, each movie is the only one in its class → Overfitting!
- Can we get the machine to learn from the data which are the relevant classes and how many we need?

Restricted Boltzmann Machine (RBM)

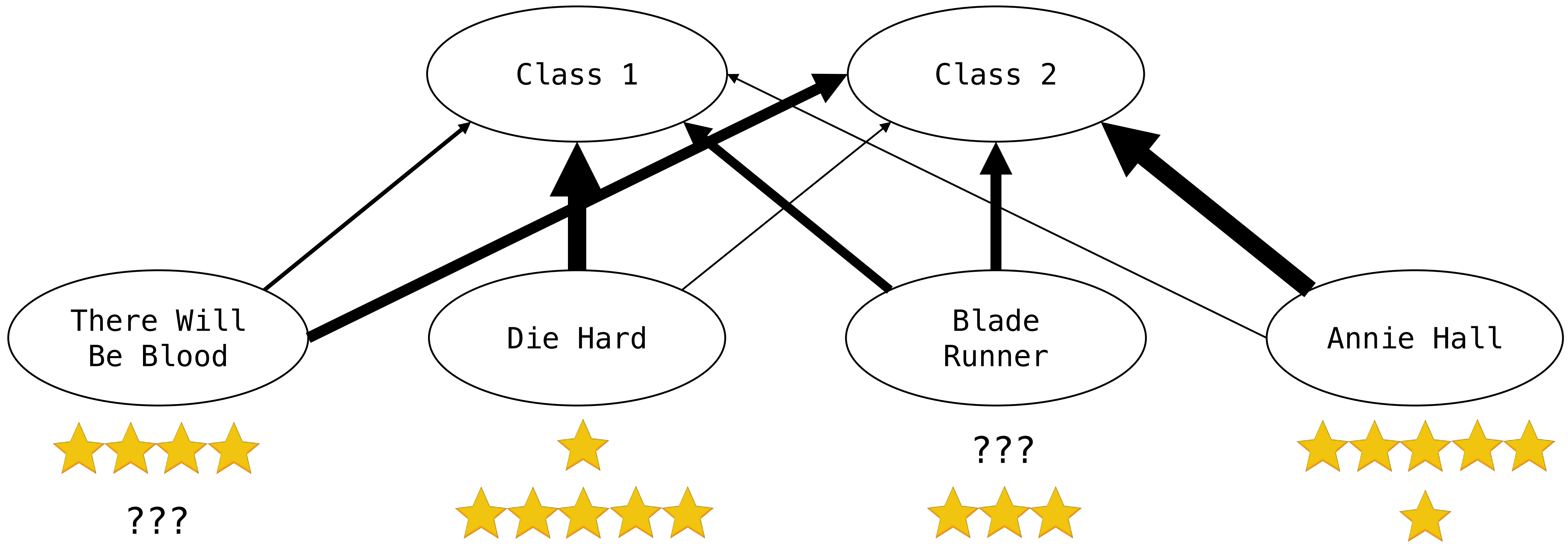
Model



- Each edge is a connection between two neurons, edge has a weight: strength of the connection.
- Training: Learn the weights that best fit the data from the ratings (black-box for now).

RBM (after training)

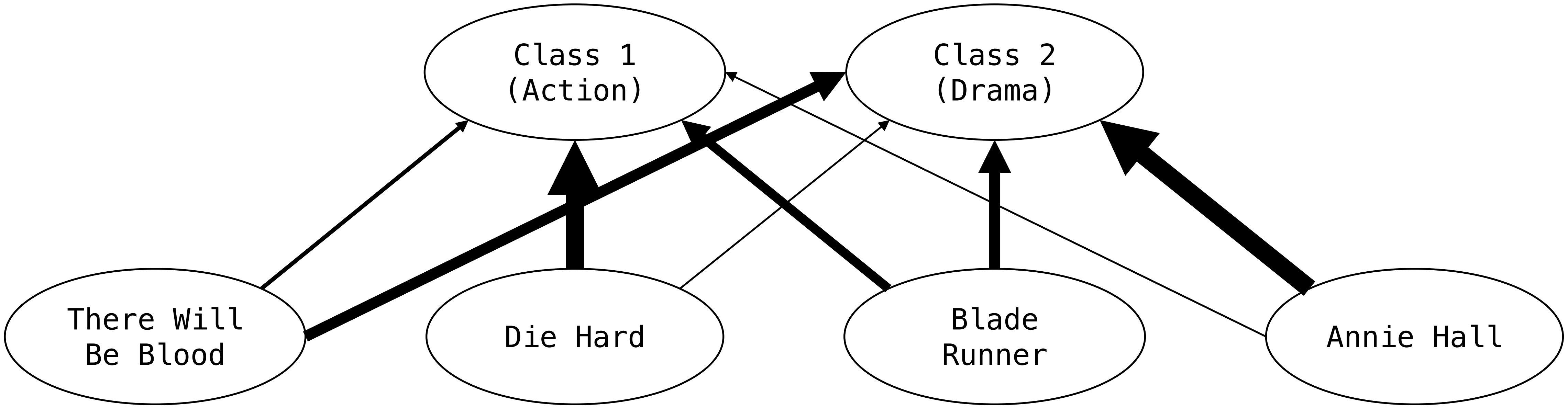
Train



- After training, the algorithm decides that the edge strengths above fit the data best.
- The algorithm uses the two upper neurons as “empty vessels” for discriminating the movies.

RBM (after training)

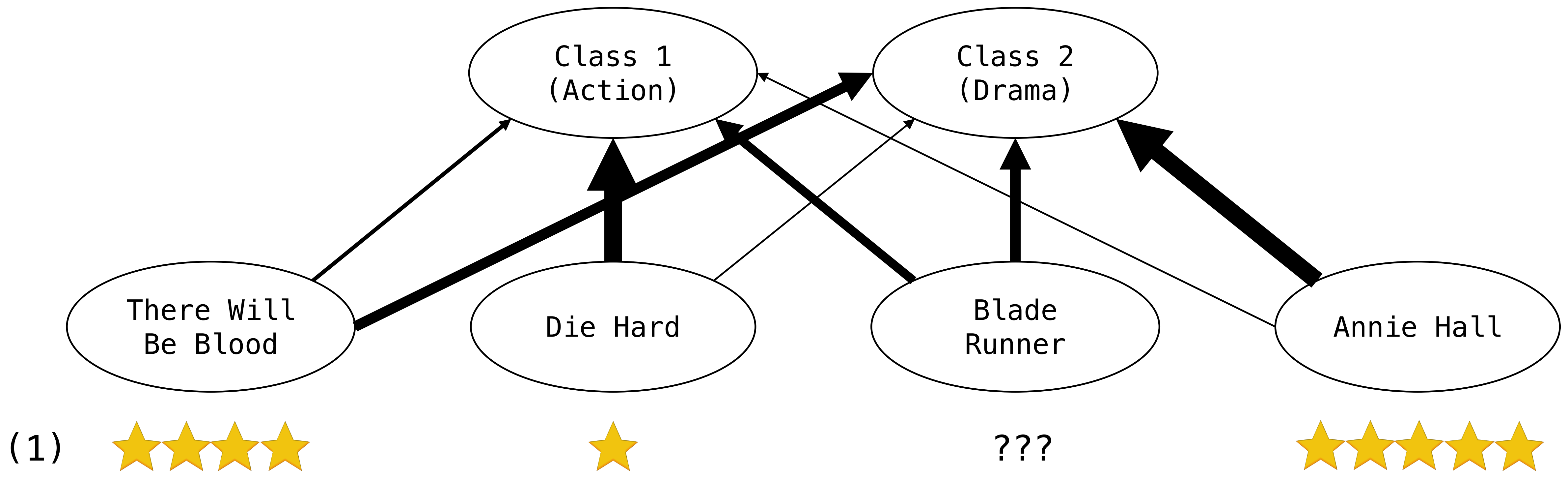
Train



- We understand the classes are “Action” and “Drama”, though the machine does not know that.
- Next step: predict the missing ratings.

Will Alice enjoy Blade Runner?

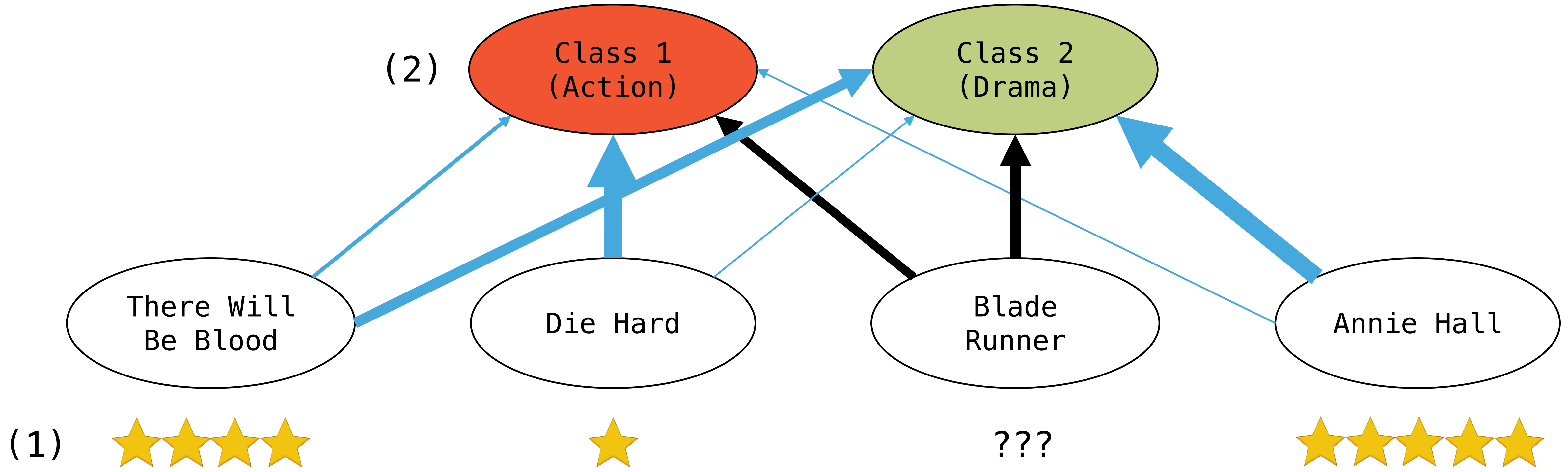
Predict



› First we plug Alice's ratings to the bottom layer.

Will Alice enjoy Blade Runner?

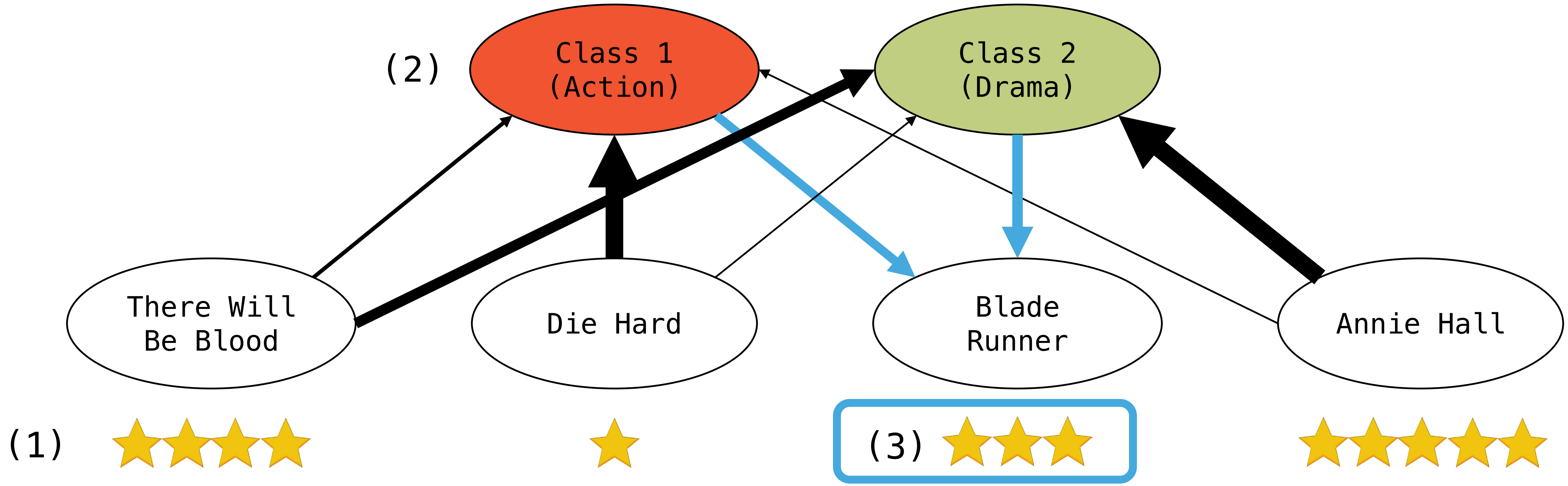
Predict



- The inputs are propagated along the colored edges, and activate neurons on the upper layer.
- Alice does not enjoy action movies so much, so the Action neuron is poorly activated.

Will Alice enjoy Blade Runner?

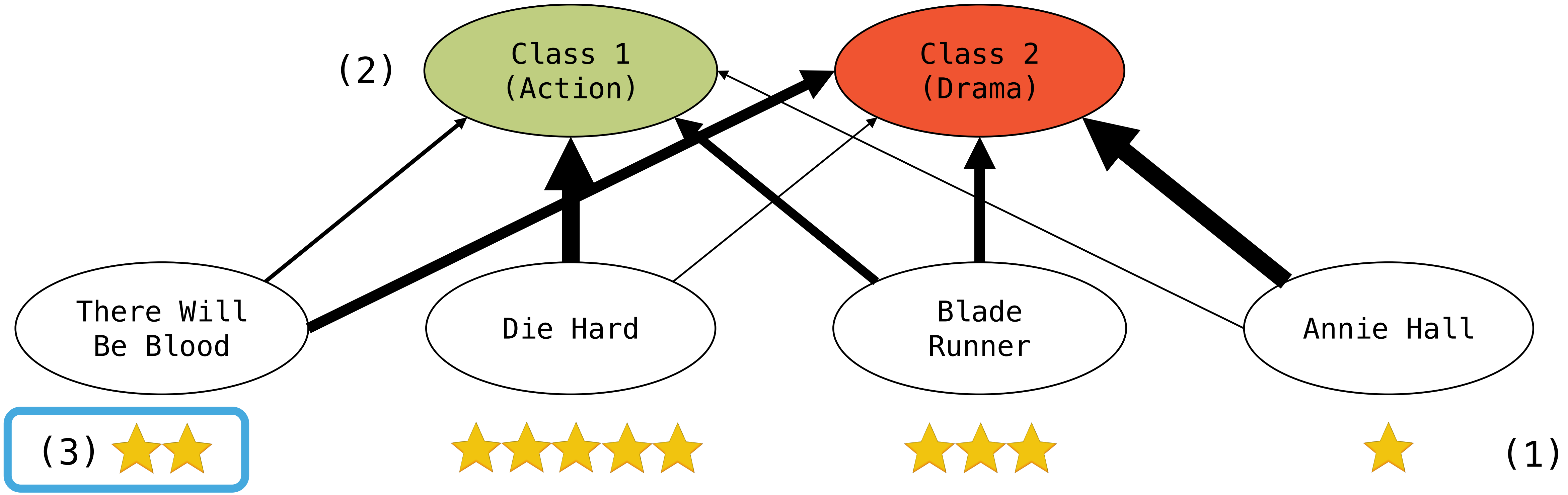
Predict



- But Blade Runner is mostly a drama, so the Drama neuron is giving a lot of points!
- The final rating will reflect both sides by averaging the activations along the edges.

Will Bob enjoy There Will Be Blood?

Predict



› Bob is the opposite, really enjoys action movies, so the Action neuron is highly activated.

› Since There Will Be Blood doesn't have so much Action, we predict Bob will give a poor rating.

Postmortem

RBMs extract features from the movies by understanding the similarities

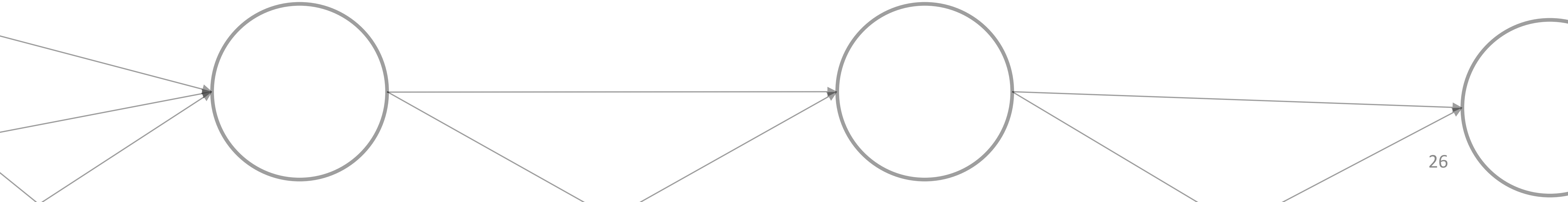
- RBMs are an example of Sparse Distributed Representation: we tag a movie with labels that belong to a set much smaller than the set of movies, e.g. Action, Drama, French …

- In essence, we map the very large space of movies into a smaller set of discriminating features. Deciding how many features are needed can be automated (see testing and cross-validation, not covered today).

- We do not tell the machine what each of these features should be: the machine will use them in a way that discriminates better, but doesn't know what they correspond to in the "real" world. Assumes user ratings are "consistent".

- There is ongoing work to automate that process too, so that machines can sensibly classify large collections by themes that they are able to extract (e.g, read news articles and know the dominant theme of article *x* is "Economics").

Contents of the talk



Neural networks: Our dataset

- N pictures tagged either `0` `(Cat)` or `1` `(Dog)`.
- All pictures are black and white, and measure `100 x 100` pixels.
- Pixels have a value between `0` and `1` depending on their intensity (`0` = black, `1` = white, `0.5` = grey).

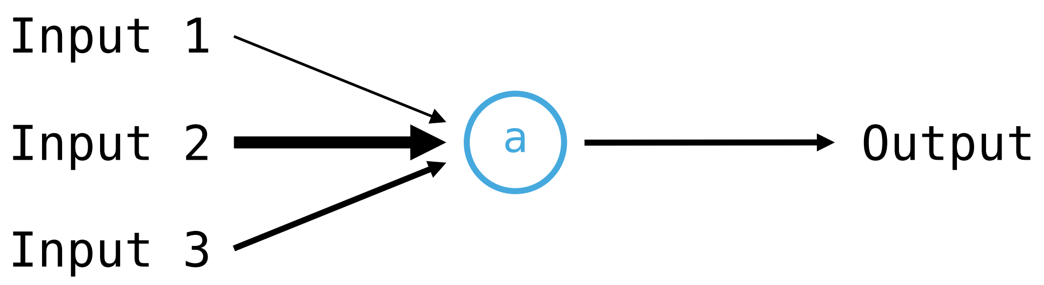
0

1

A closer look at the neuron

Small unit of computation, transforms inputs into an output

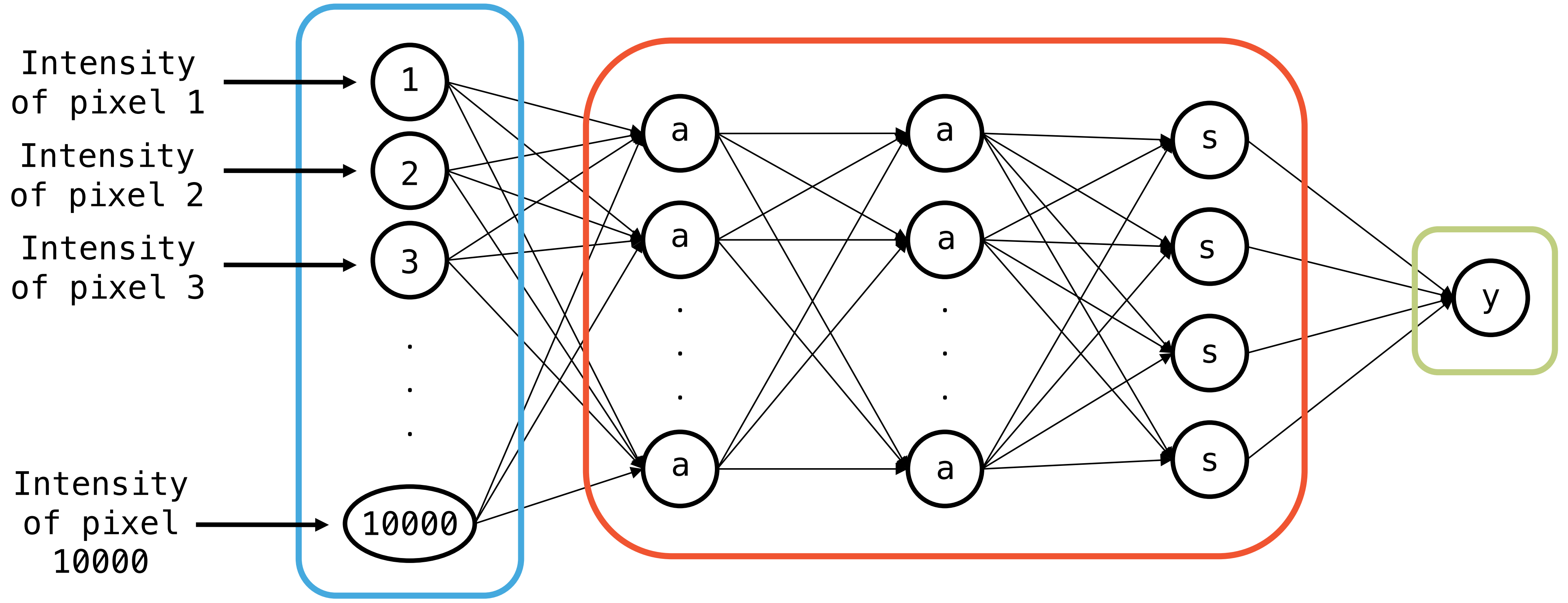
- Neural networks are based on neurons: they receive weighted inputs, activate, and output a value.



- a can be any function that maps a number to another number, but in practice some work better than other. Why? We are not sure...

A neural network

Model



Input layer

Hidden layer

a,s: activation functions,
e.g a(x) = max(x,0)

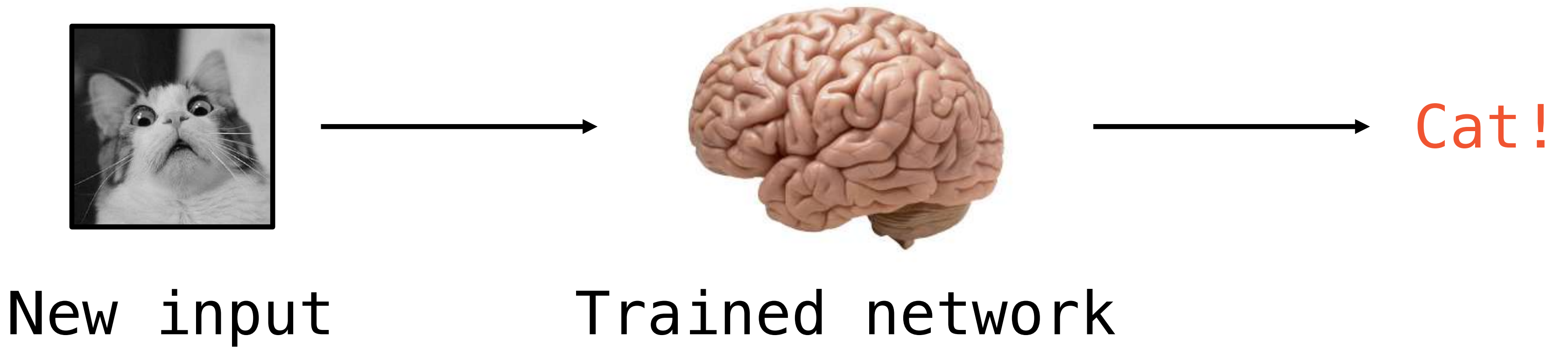
Output layer

y: output,
between 0 and 1

Neural networks: Intuition

Transform the input into a decision variable using weights obtained from training

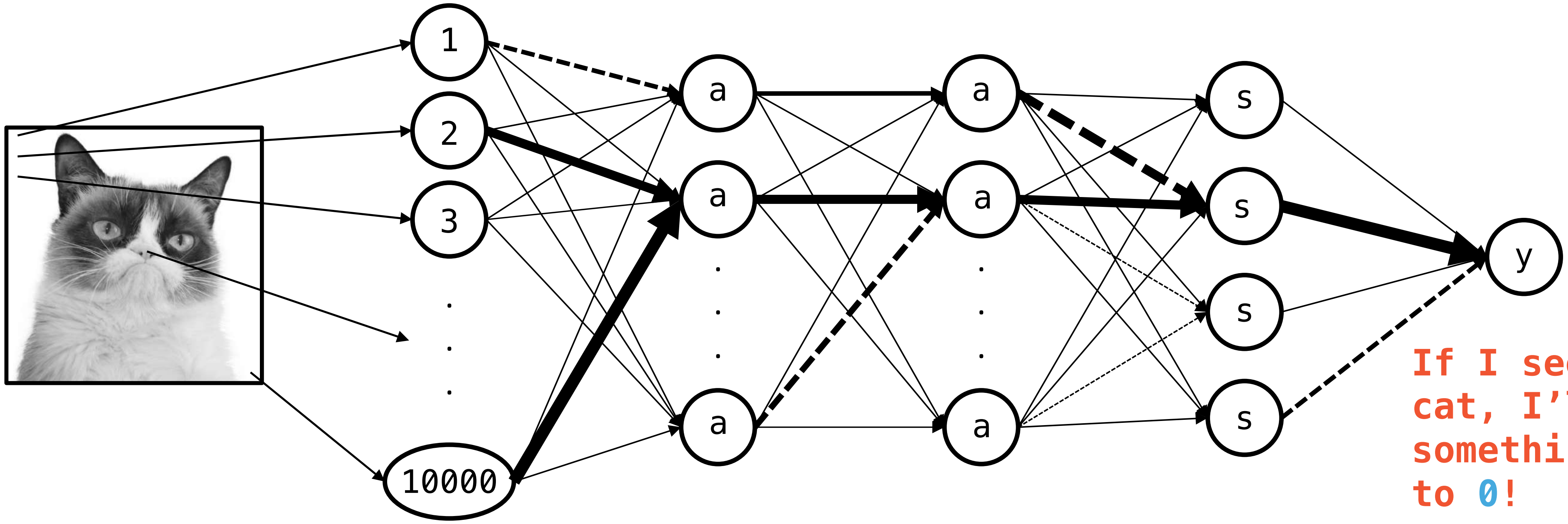
- First we train a neural network with our collection of data: learn the best edge weights that represent the data (similar to the RBM / movie ratings example).
- Second, we can input new data and ask the neural network "is it a Cat?"
- The larger our network is, the higher its capacity to learn (but it may overfit).
  Just as with the RBMs, we want to find the sweet spot.



Neural network training

Train

- We start training the network by showing it images with their tags.
- The net reacts by updating the weights on its edges so that it outputs something close to the tag.
- Edges can react positively (plain lines) or negatively (dashed lines).

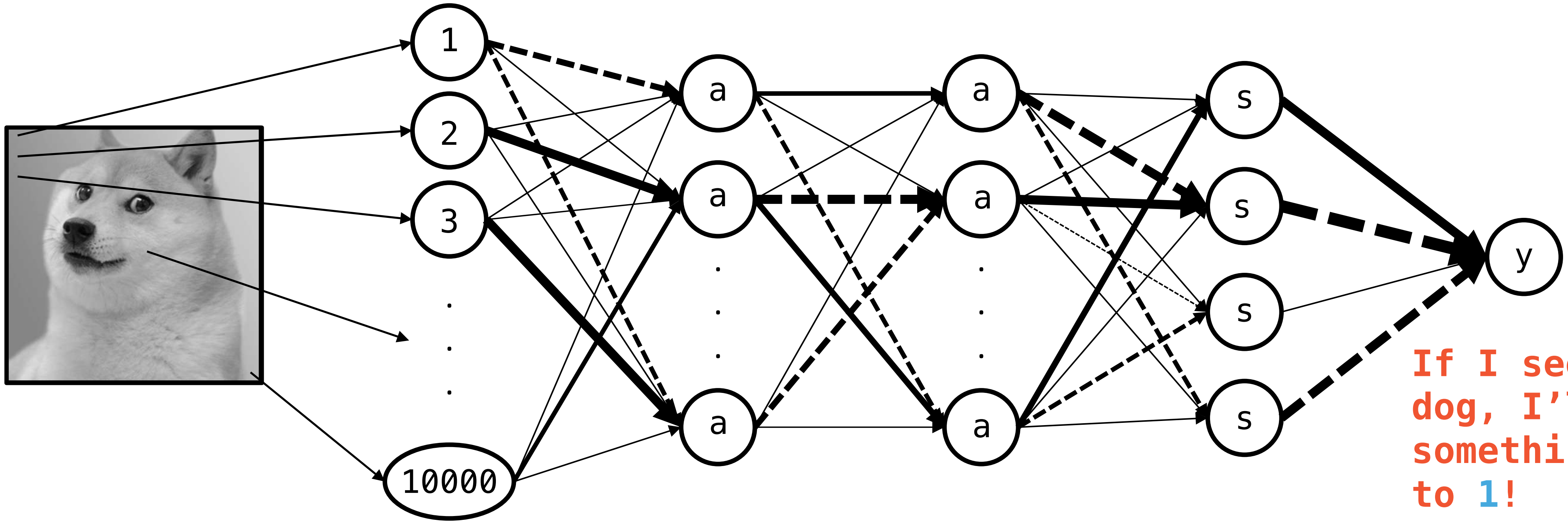


If I see another cat, I'll say something close to 0!

Neural network training

Train

- As we show more and more images to the net, the weights tend to change less.
- After a while, the net is trained to output something close to `1` if the input is a dog picture, or `0` if it is a cat picture.

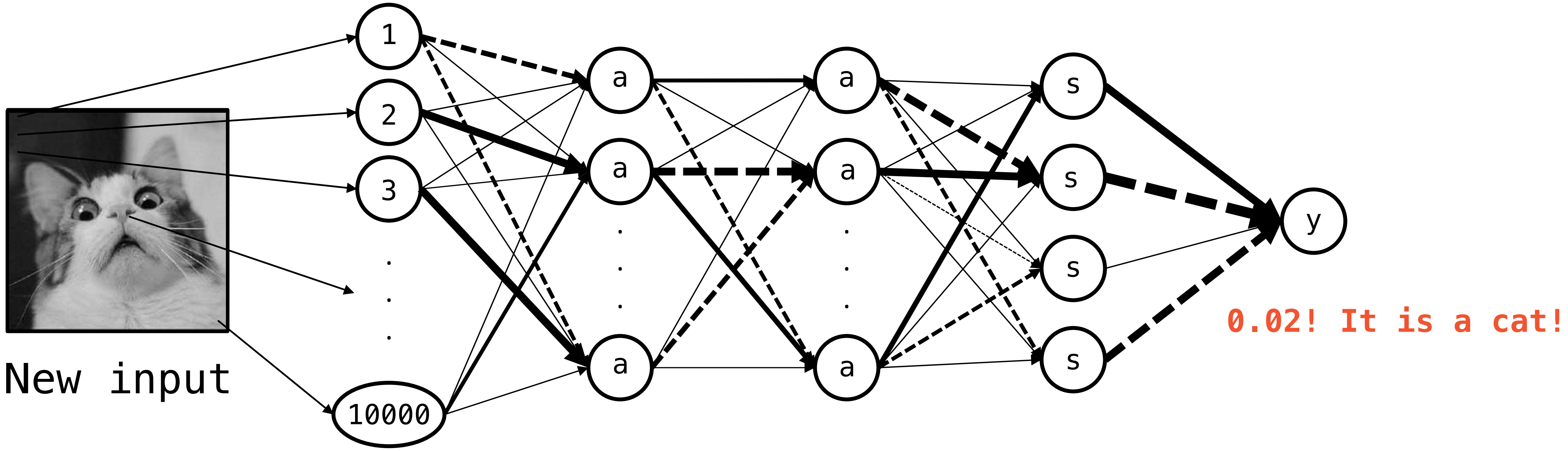


If I see another dog, I'll say something close to 1!

Neural network training

Predict

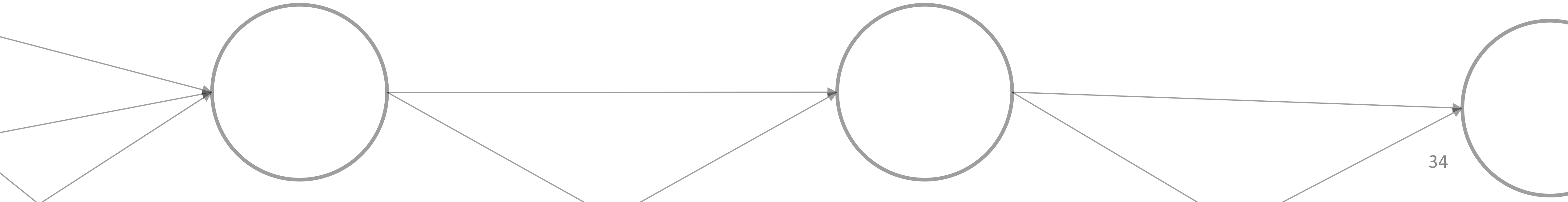
- Once the network is trained, we can show it new unseen inputs and classify them using the output.
- If the performance of the network is not good, we can try training it again with more layers or more nodes.



Caveats

Neural networks are hard to train: need lots of data, fast processors and a bag of tricks

- The larger the nets are, the better capacity to learn they have. This lead to thinking the model was pretty poor because the processing power in the 90's was not good enough to train large nets.
- Turns out it works "unreasonably" well, but it needs to be large enough and so requires good GPUs or custom chips to be efficient. Today: deep learning, a lot of hidden layers.
- Each neural network is different! Possible to customise them to work on particular tasks (see convolutional neural networks for image processing).



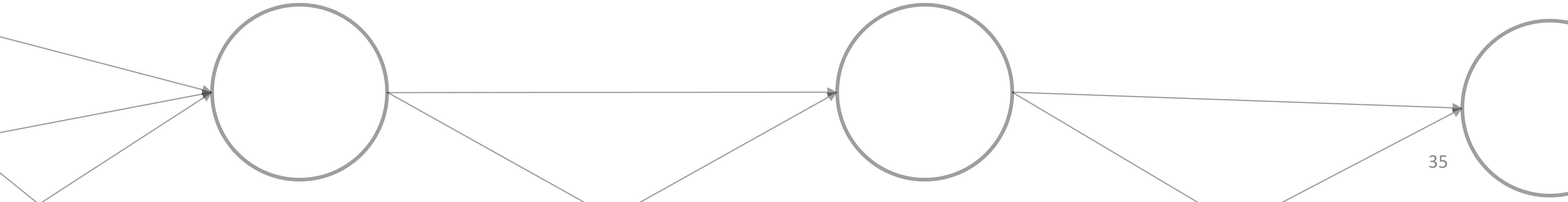
Contents of the talk

Machine Learning basics

- The Data zoo: Supervised / Unsupervised
- Modeling, training, predicting

The Model zoo

- Collaborative filtering: Predicting a user rating
- Neural networks and deep learning: Dog or cat?



Deciding where to start

ML offers a range of models suited to very different data types and problems

- ML is transforming the way we think about problems: sometimes the solution is just too complex to find the rules that governs it (what makes a dog? how are ratings given?)
- First step to successfully use ML: identify such a complex problem. Is there enough regularity? Can I use data to infer patterns?
- Second step: what does the data look like? This will influence the type of models you will use. Is it labeled? Can it be labeled? (by using online labor markets, e.g Mechanical Turk)
- Third step: how to implement my model? Can I use an off-the-shelf implementation? Look at scikit-learn for Python, the MLlib of Apache Spark, TensorFlow, all easy to prototype.

Where to go from here

- We have seen how different models are set up and used for prediction, but we were a bit evasive about the training part (how do we get the *best* weights?)
- There is a rich literature on how to train networks, e.g gradient descent algorithms, regularization…
- There are also a lot more models out there tailored to particular tasks, e.g recurrent neural networks are great at data that is sequential, such as speech data (predicting the next word), sensor data (predicting missing data)…
- ML is a vibrant field, constantly improving, transforming all disciplines. Better to keep a look out for what's next!