



311 Introduction to Machine Learning

Summer 2024

Instructor: Ioannis Konstantinidis

Overview



- Feature engineering vs. feature learning
- Error backpropagation
- Stochastic Gradient Descent

Feature Engineering

accepting (word
article).
focus n point
converging rays of light,
heat, waves of sound, meet;
centre of activity or
adjust; cause to converge;
concentrate; a focal
pertaining to focus

Features: domain knowledge

RAW DATA

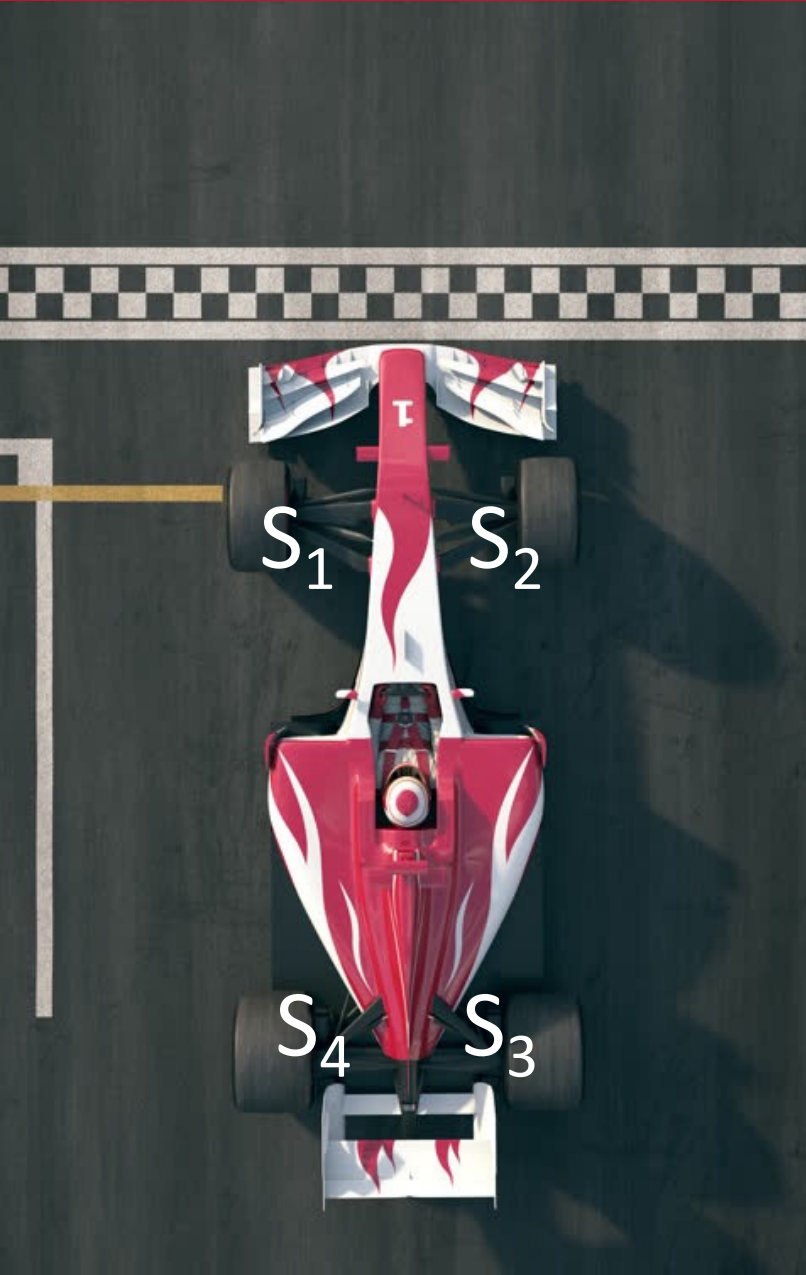
Four sensors measuring rotation speed (spin)
at each wheel: S_1, S_2, S_3, S_4

NEW FEATURES

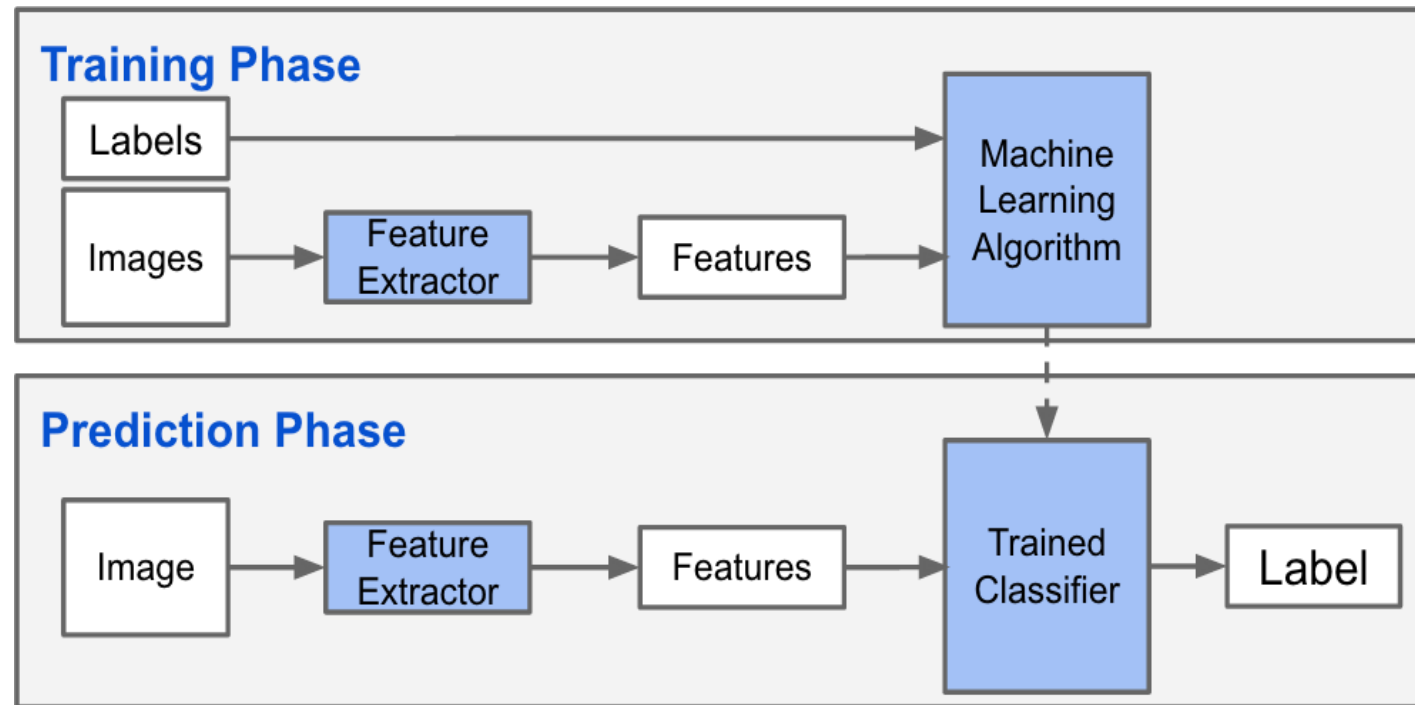
$$T_1 = \left\{ \left(\frac{S_2 + S_3 + S_4}{3} \right) - S_1 \right\} / 2 = -\frac{1}{2} S_1 + \frac{1}{6} S_2 + \frac{1}{6} S_3 + \frac{1}{6} S_4$$

EXPERT KNOWLEDGE

If a feature starts to veer away from zero, then a tire is
spinning faster than the others (possible flat)

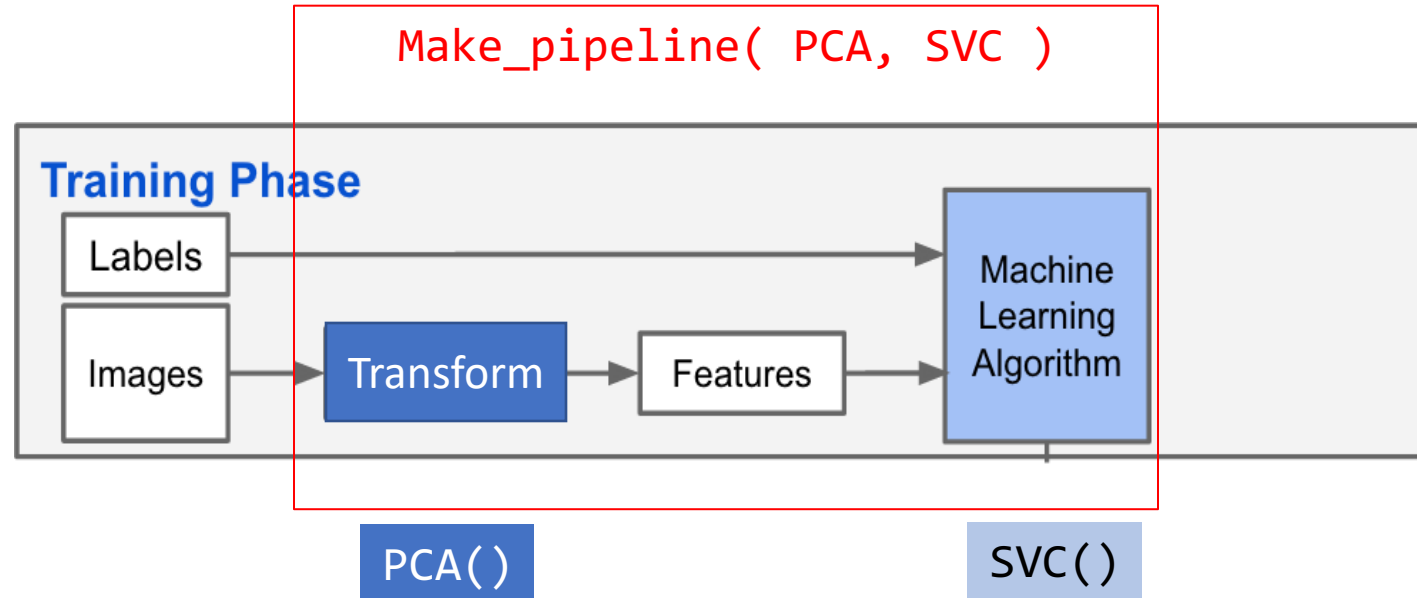


Feature extractors help unscramble the features from the raw data, and prioritize features for selection



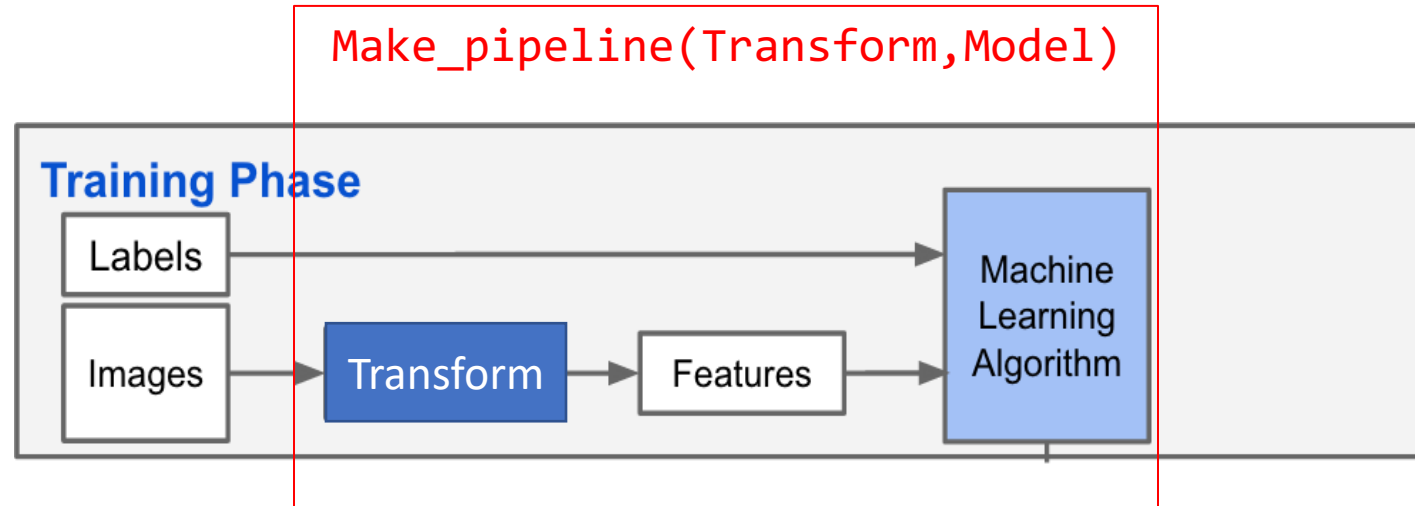
Machine Learning Phases

Feature engineering example: PCA



PCA is most commonly available data transform, because it is the most generic

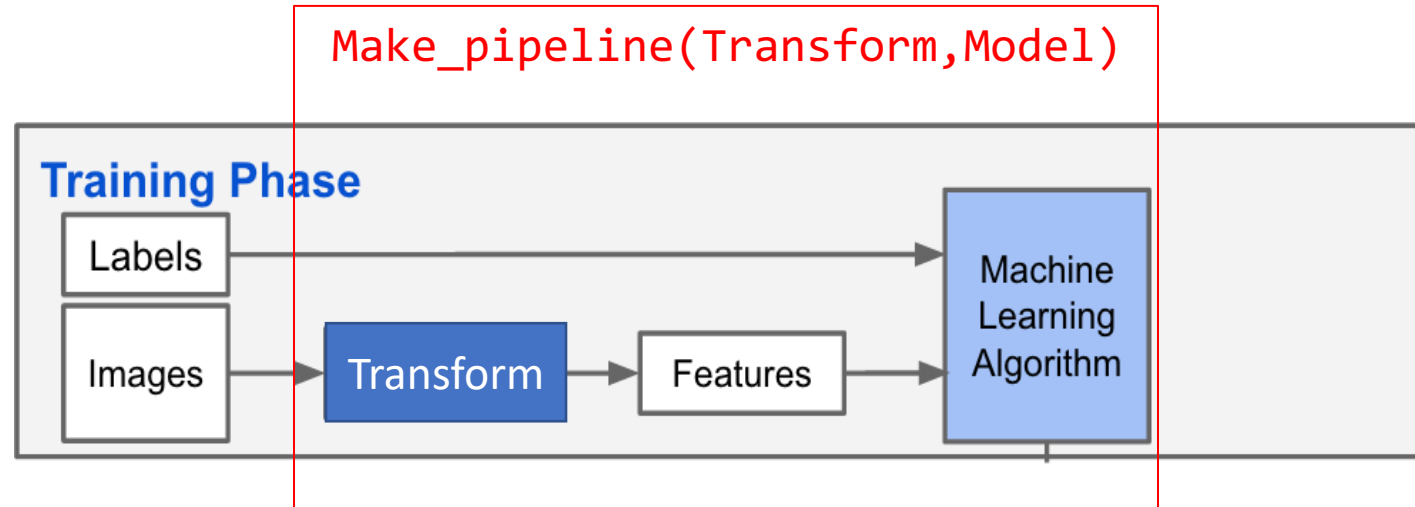
Feature engineering: a lot of possibilities



PCA is most commonly available data transform because it is the most generic

There are many other choices

Feature engineering: a lot of possibilities

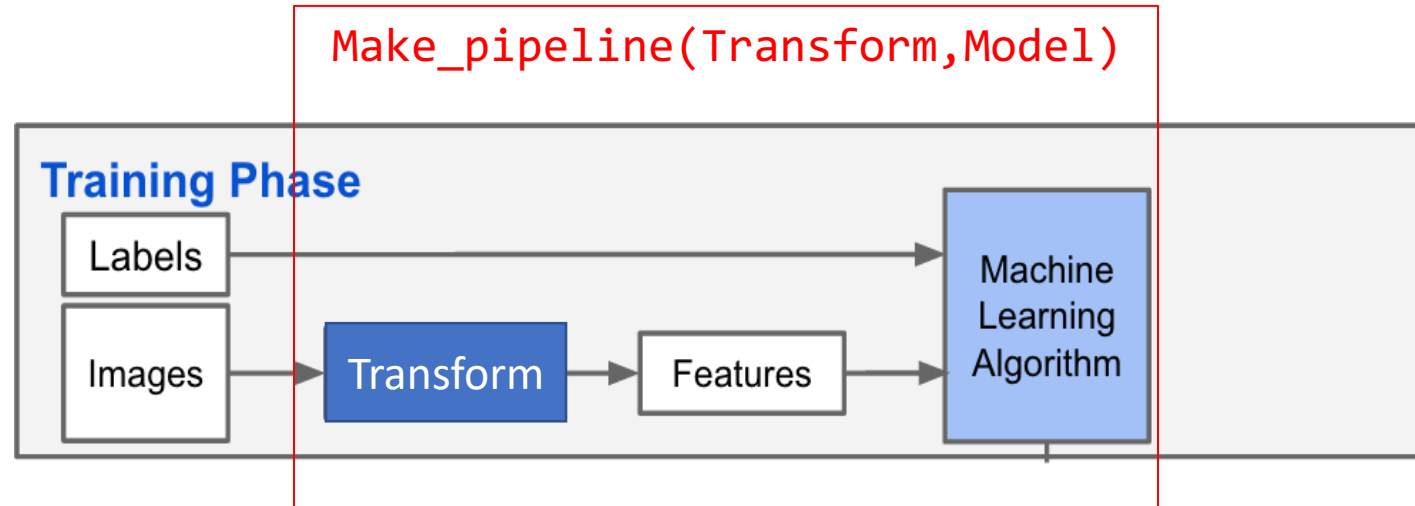


PCA is most commonly available data transform because it is the most generic

There are many other choices:

- Fourier Transform: extract frequencies from wave signals

Feature engineering: a lot of possibilities

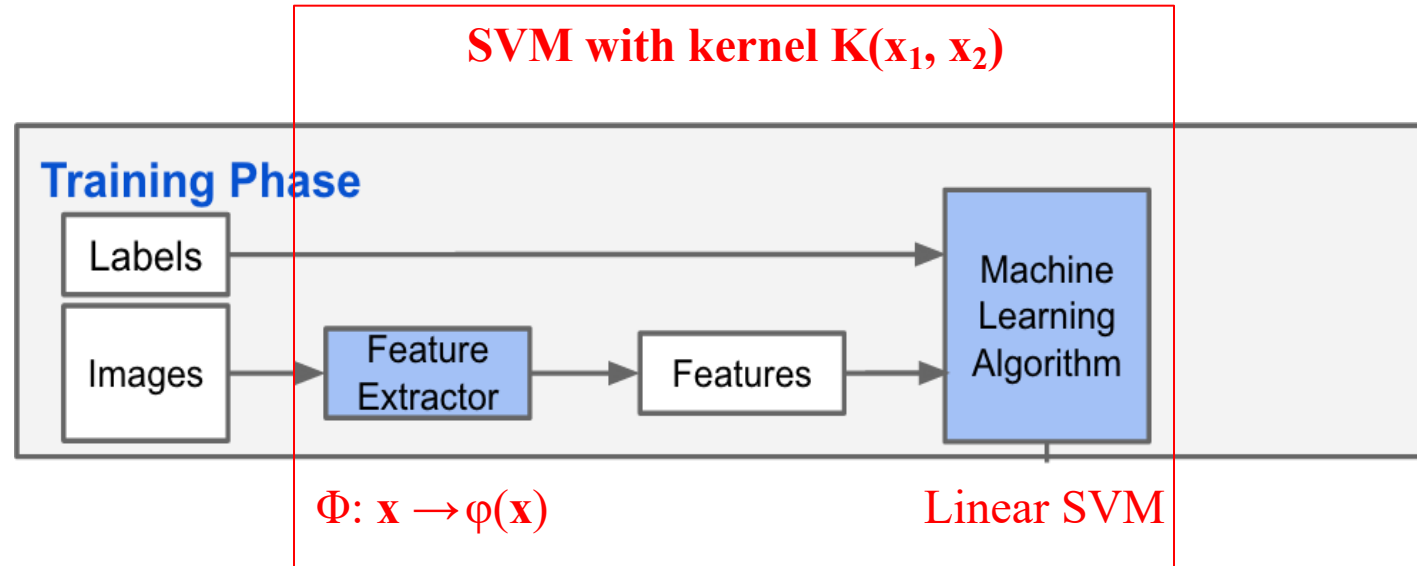


PCA is most commonly available data transform because it is the most generic

There are many other choices:

- Fourier Transform: extract frequencies from wave signals
- Wavelet Transform: extract levels of detail from images

The kernel trick masks a data transform



Think of

```
SVC( kernel='rbf' )
```

as being the same as

```
make_pipeline( rbfTransform, SVC )
```

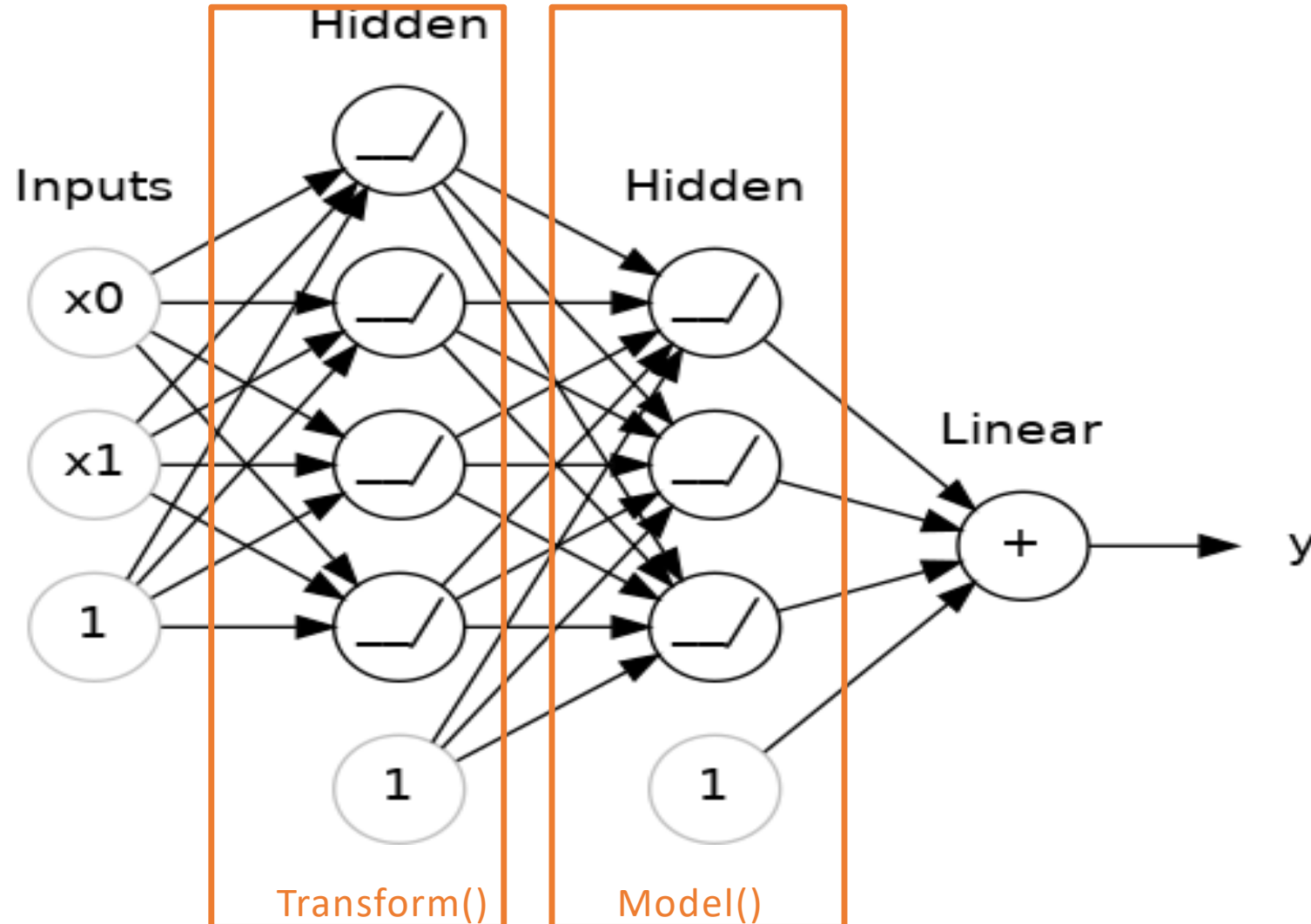
Feature engineering: drawbacks

- Feature engineering is difficult, time-consuming, and requires domain expertise.
- It is in the spirit of symbolic AI, instead of the modern connectionist paradigm

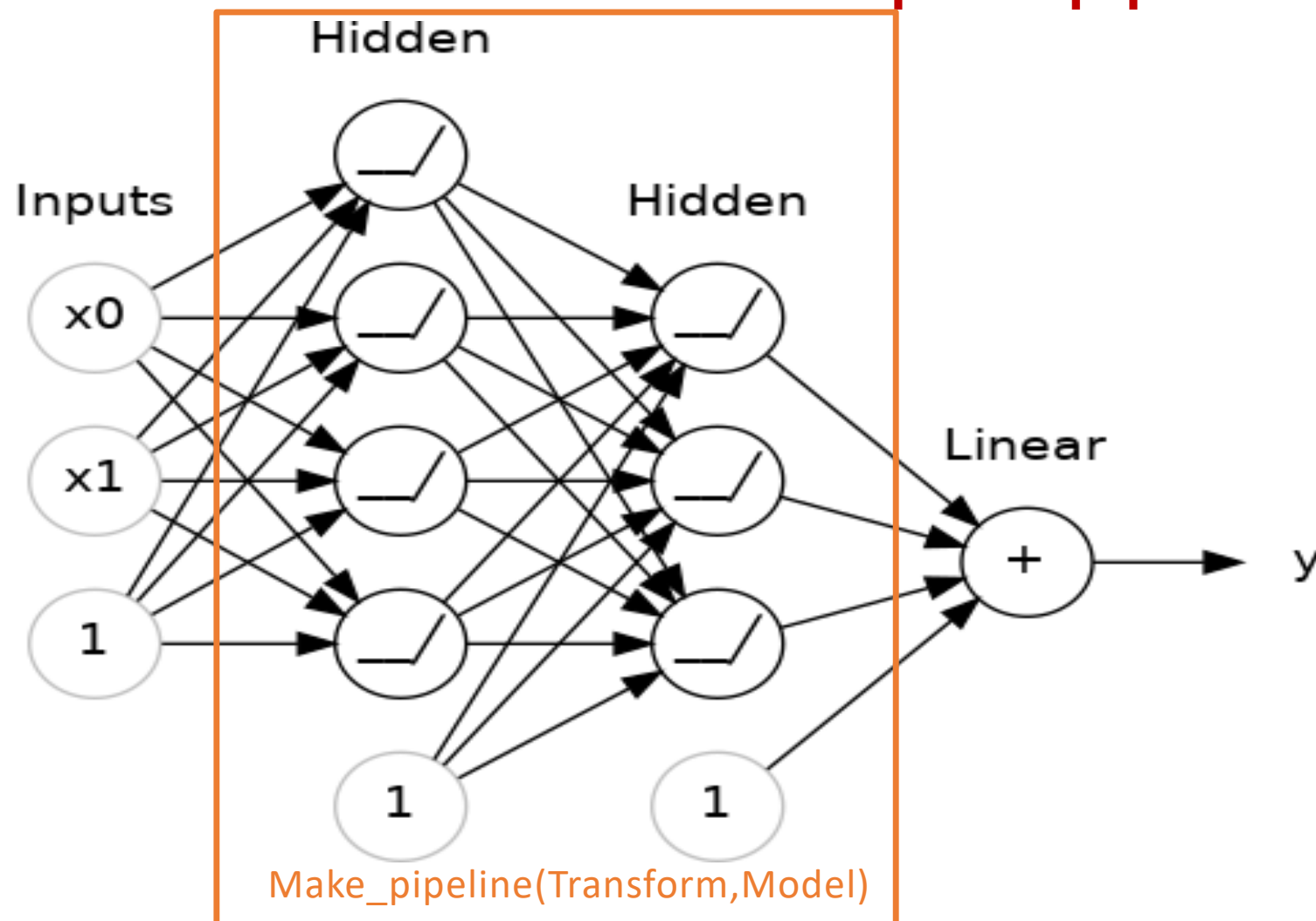
Feature Learning

accepting (word
article).
focus n point
converging rays of light,
heat, waves of sound, meet;
centre of activity or
interest; pl focuses, foci; v
adjust; cause to converge;
concentrate; a focal
pertaining to focus

Multilayer Neural Networks are a complete pipeline



Multilayer Neural Networks are a complete pipeline



Multilayer Neural Networks are a complete pipeline

- Feature engineering is automatically “baked into” the process
- Initial layers pick out “low level” features
- Later layers process these transformed data to compute “higher level” features
- “Top” layers perform classification tasks based on these custom-designed features

Hierarchical representations

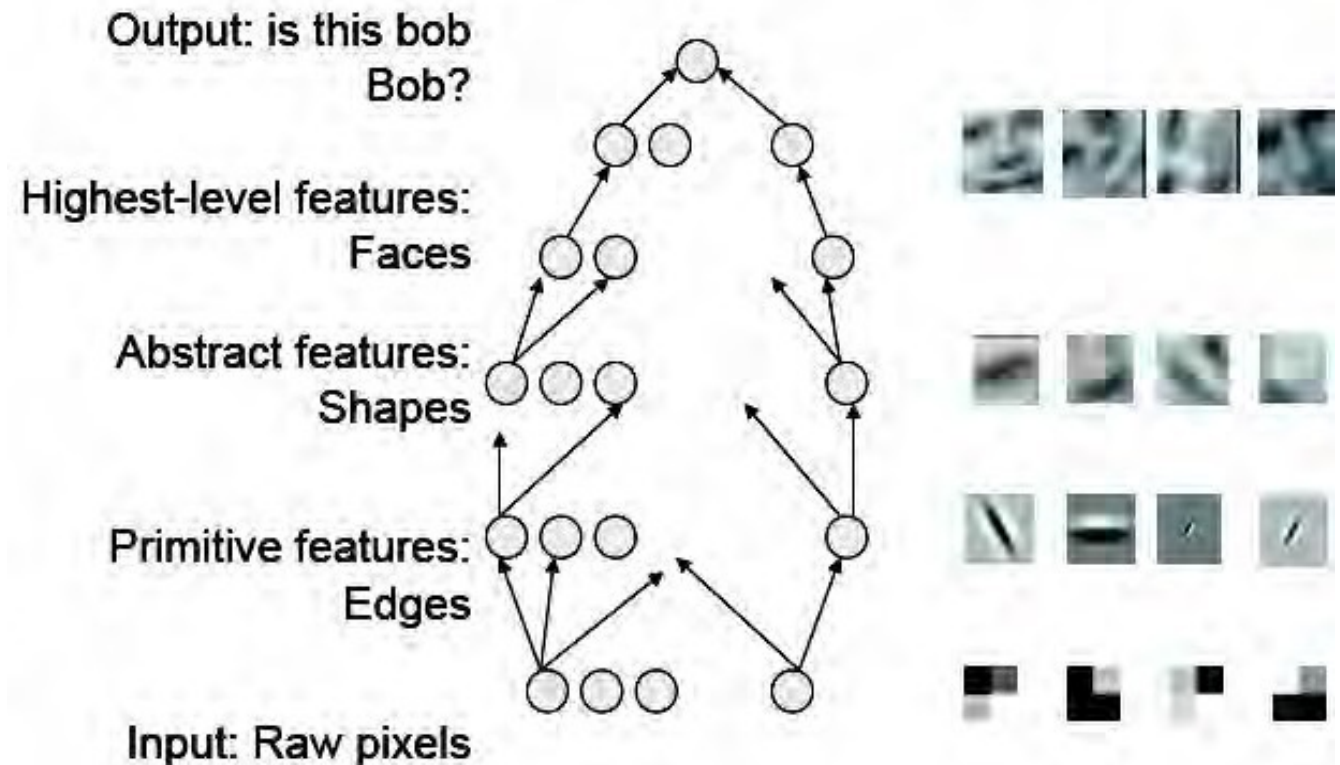
“Deep learning methods aim at learning feature hierarchies with features from higher levels of the hierarchy formed by the composition of lower level features.

Automatically learning features at multiple levels of abstraction allows a system to learn complex functions mapping the input to the output directly from data, without depending completely on human-crafted features.”

[Bengio, “On the expressive power of deep architectures”, Talk at ALT, 2011]

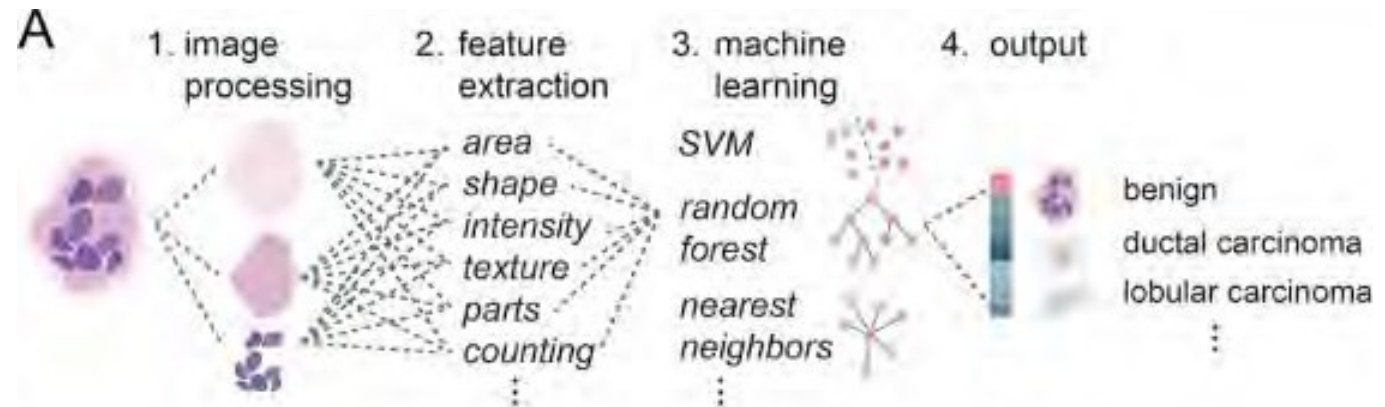
[Bengio, Learning Deep Architectures for AI, 2009]

Deep learning architecture

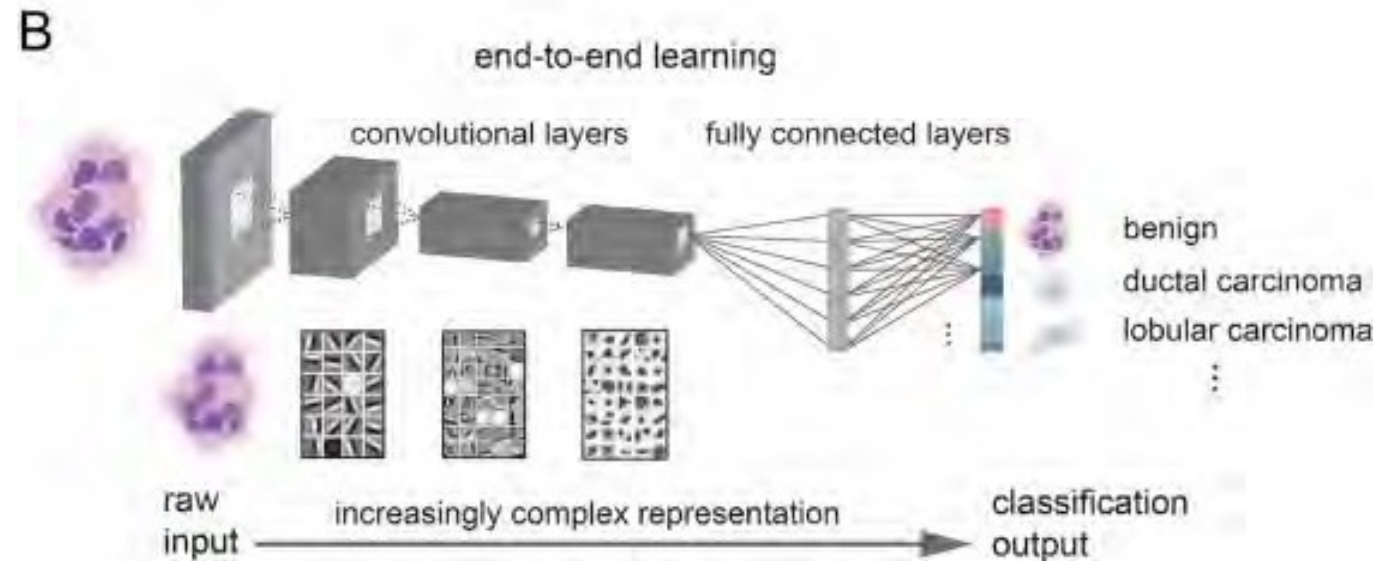


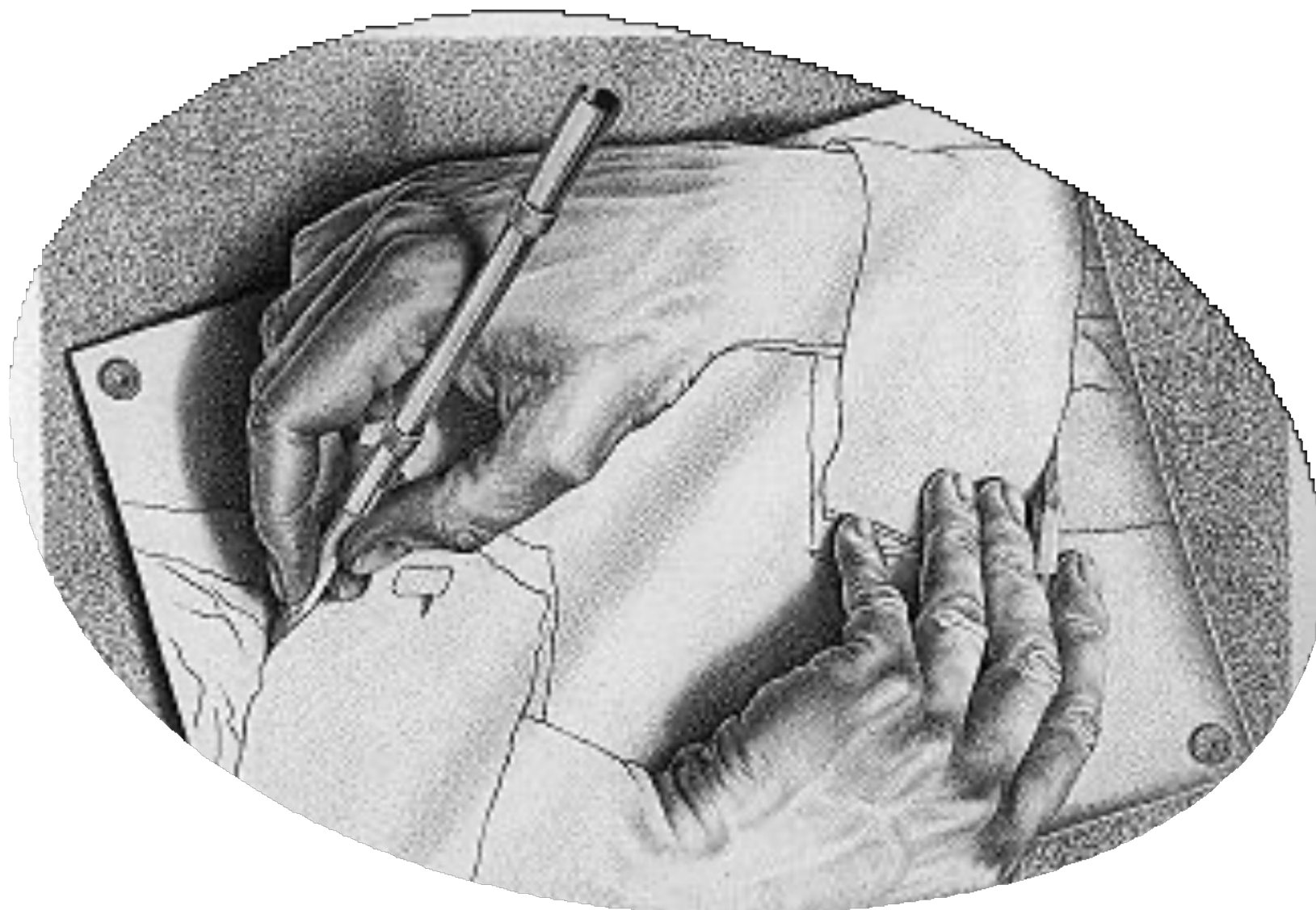
The Deep Learning Revolution

Earlier ML

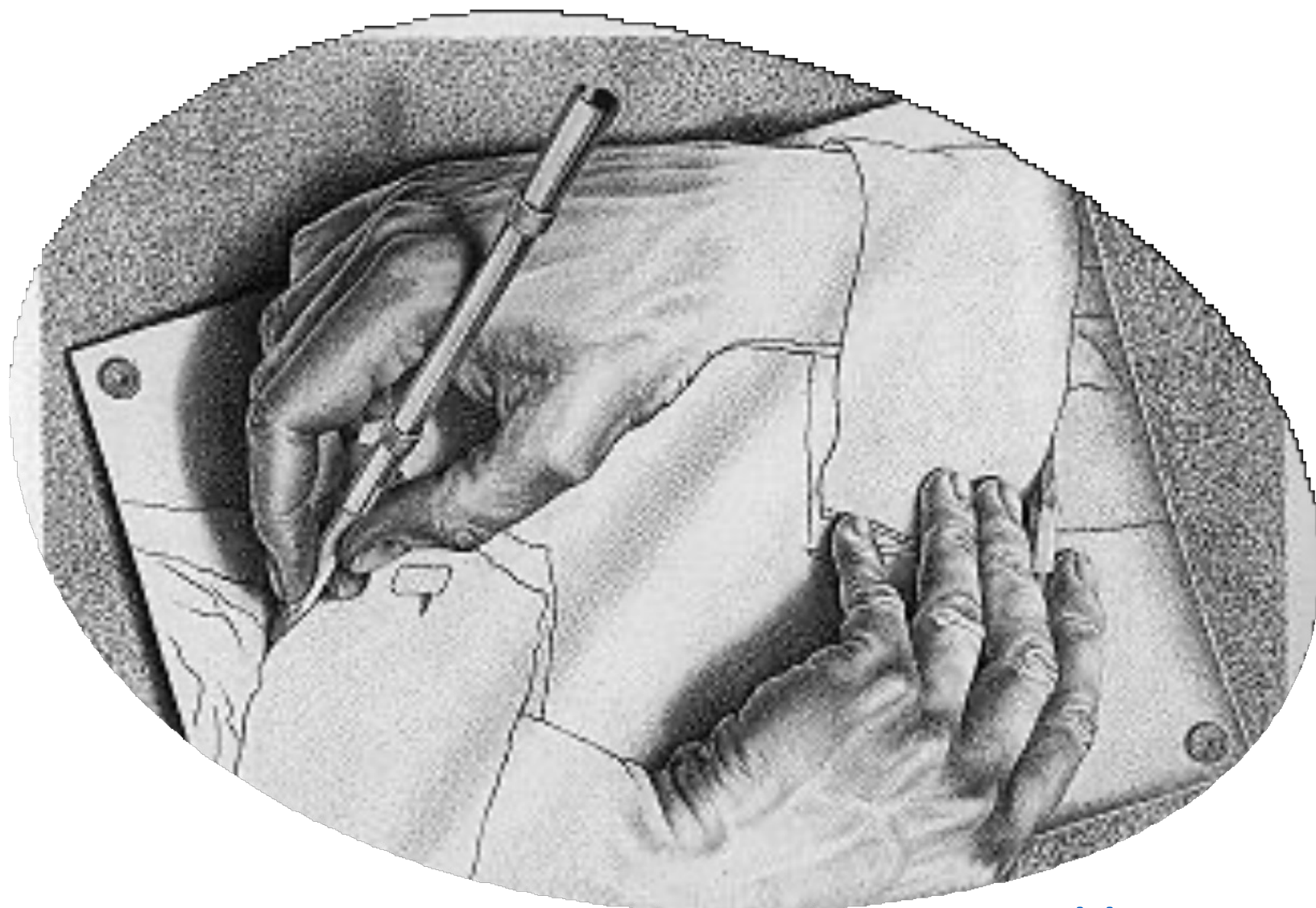


Deep Learning





Hands-on Example: MLPClassifier



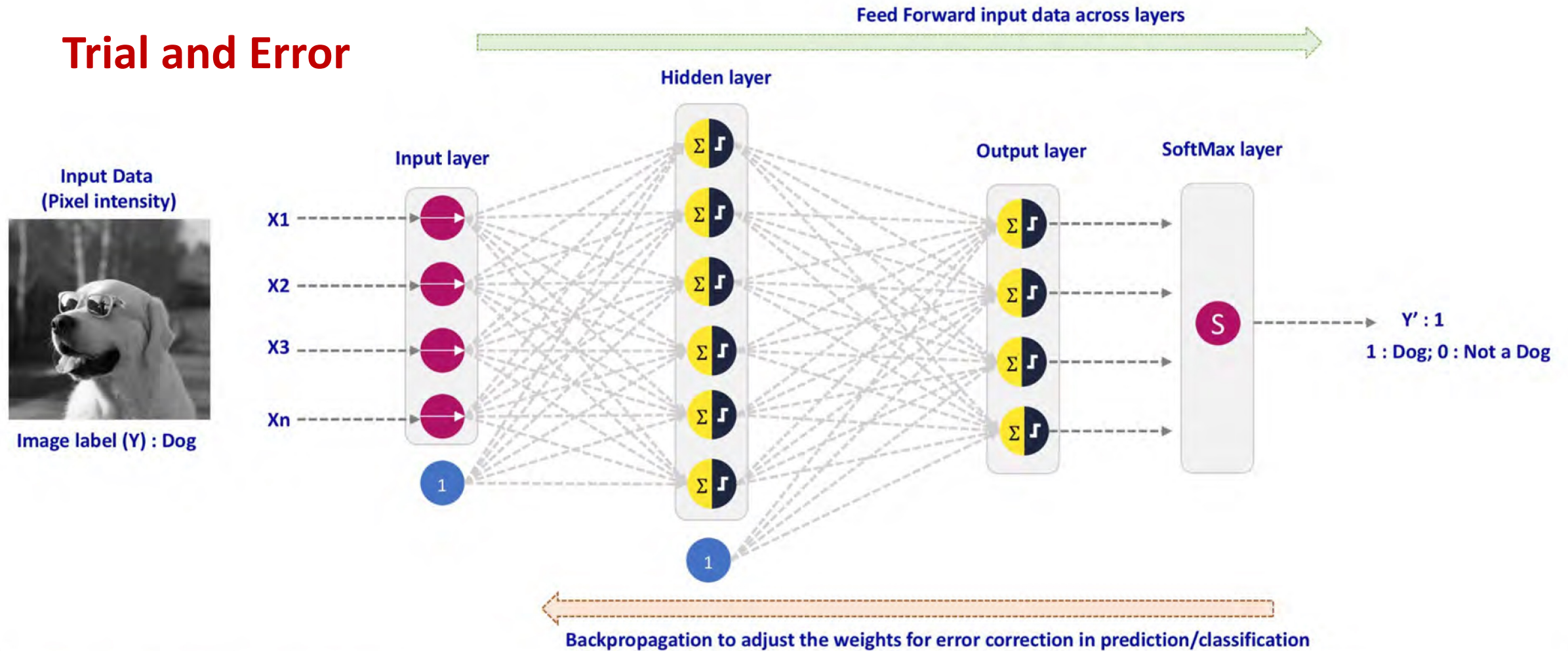
Hands-on
Example:
TF playground


<https://playground.tensorflow.org/>

But how does a neural network learn?





Trial and Error



 **Input node:** It can be a simple passthrough node or could be a transformation node (an encoder for categorical variable or a transformer for the continuous variable)

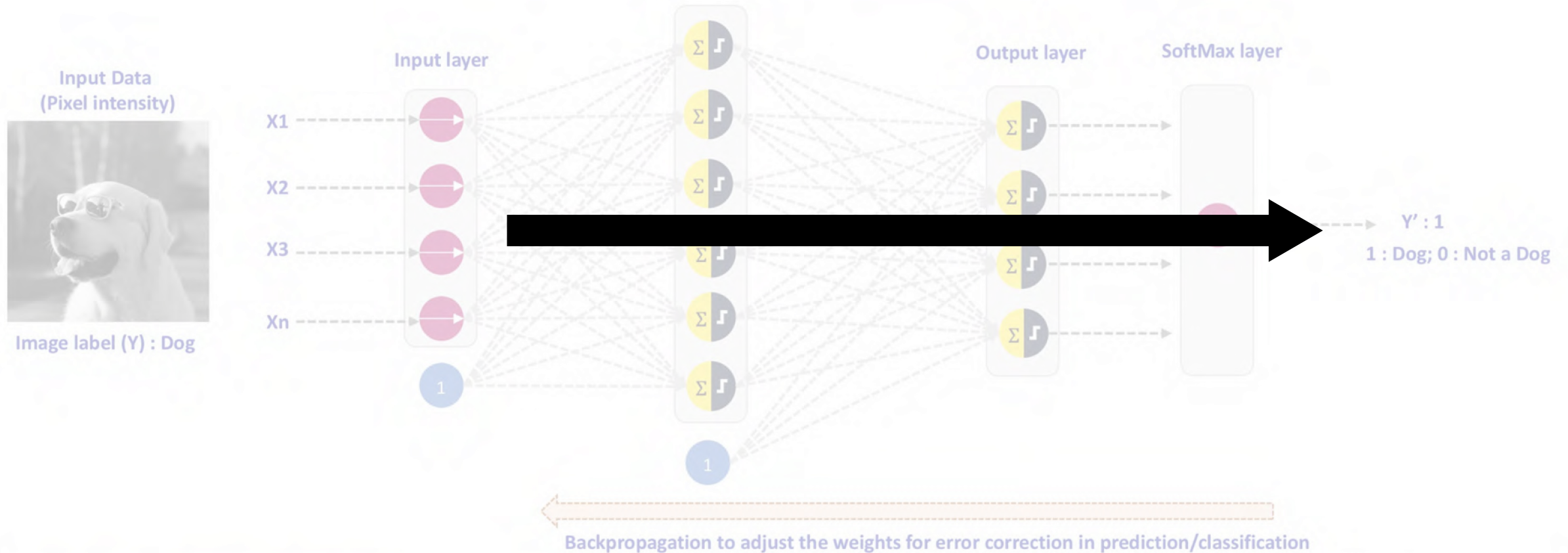
 **Bias term:** Bias term of 1 for each node

 **Neuron :** A combination of the summary and activation function; Can take any activation function

 **SoftMax :** Push the output layer values into a SoftMax for the categorization output

Trial and Error

Step 1: Use training data to make predictions



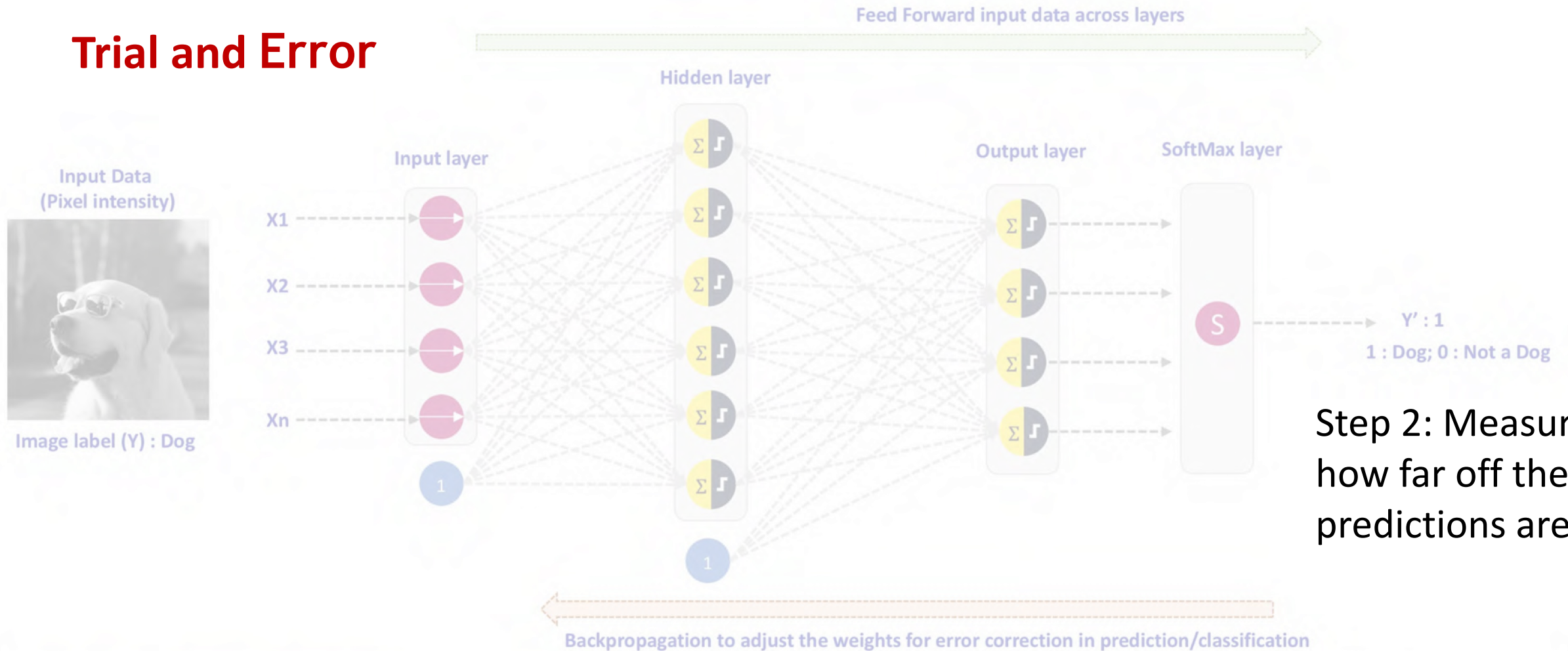
Input node: It can be a simple passthrough node or could be a transformation node (an encoder for categorical variable or a transformer for the continuous variable)

Bias term: Bias term of 1 for each node

Neuron : A combination of the summary and activation function; Can take any activation function

SoftMax : Push the output layer values into a SoftMax for the categorization output

Trial and Error



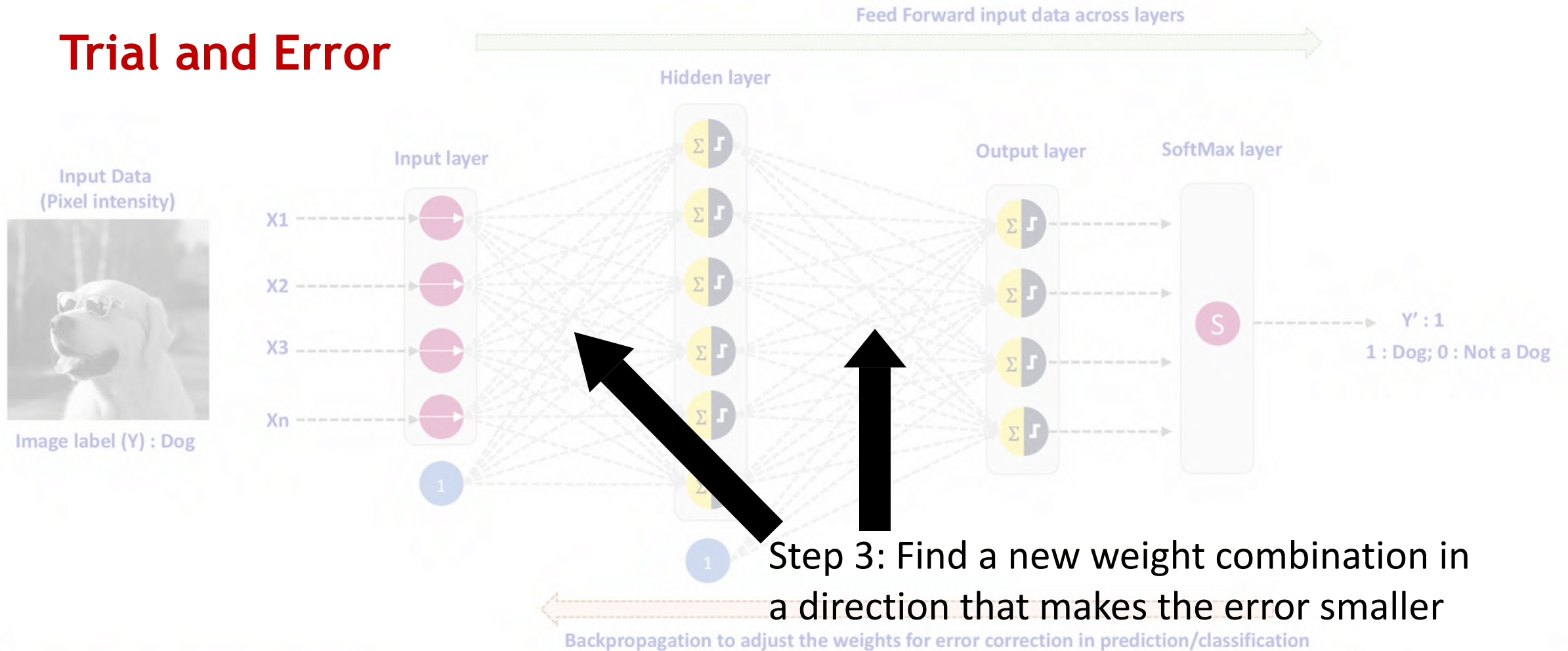
Input node: It can be a simple passthrough node or could be a transformation node (an encoder for categorical variable or a transformer for the continuous variable)

Bias term: Bias term of 1 for each node

Neuron : A combination of the summary and activation function; Can take any activation function

SoftMax : Push the output layer values into a SoftMax for the categorization output

Trial and Error



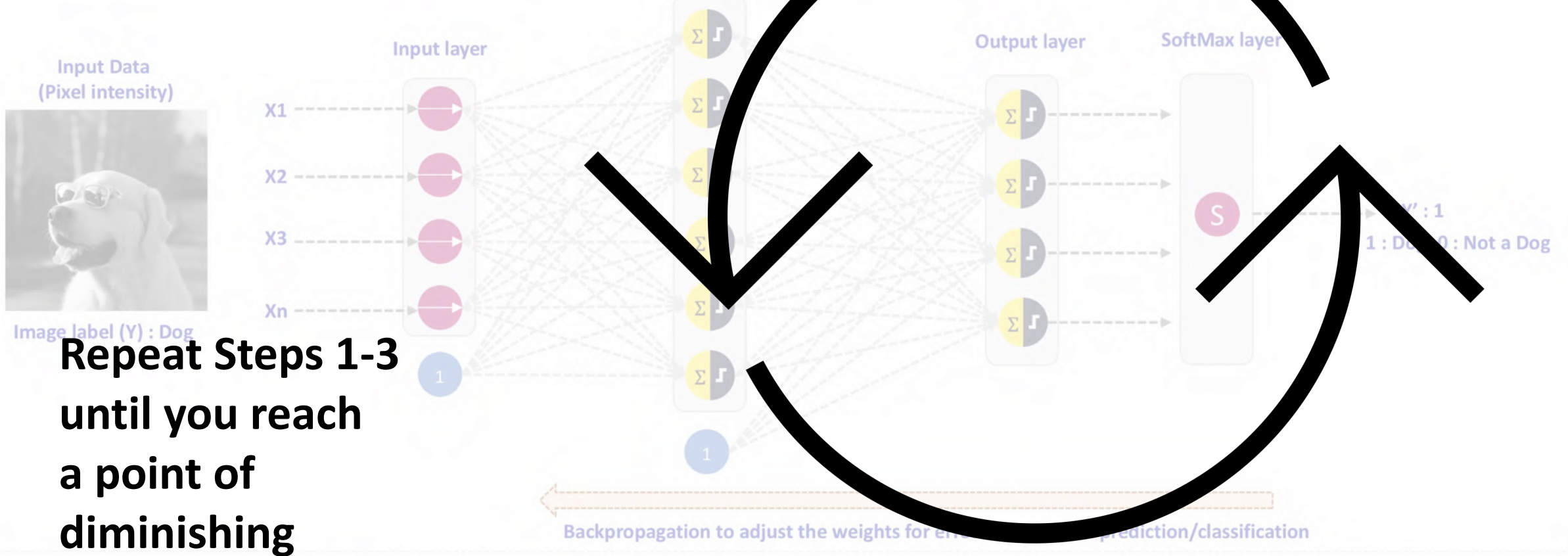
Input node: It can be a simple passthrough node or could be a transformation node (an encoder for categorical variable or a transformer for the continuous variable)

Bias term: Bias term of 1 for each node

Neuron : A combination of the summary and activation function; Can take any activation function

SoftMax : Push the output layer values into a SoftMax for the categorization output

Trial and Error



Repeat Steps 1-3
until you reach
a point of
diminishing
returns

Input layer node is a simple passthrough node or could be a transformation node (an encoder for categorical variable or a transformer for the continuous variable)

1 Bias term: Bias term of 1 for each node

Neuron : A combination of the summary and activation function; Can take any activation function

S SoftMax : Push the output layer values into a SoftMax for the categorization output

Loss functions and optimizers

accepting (word
article).
focus n point
converging rays of light,
heat, waves of sound, meet;
centre of activity or
intensity; pl focuses, foci; v
adjust; cause to converge;
concentrate; a focal
pertaining to focus

Step 2: Measure how far off the predictions are

This is the role of the objective function

- Usually called loss function when applied to Neural Networks

Least squares is a common choice: $C(w, b) = \frac{1}{2N} \sum_n \|y_n - y'_n\|^2$

- All the familiar ones apply (L2, L1, etc.)
- New choice for classification: Cross-Entropy

Regularized versions are also used (hyperparameter alpha)

Step 3: Find a new weight combination that makes smaller errors

The *optimizer*, usually a variation of stochastic gradient descent (SGD), considers each weight in turn (this can be parallelized)

Step 3: Find a new weight combination that makes smaller errors

The *optimizer*, usually a variation of stochastic *gradient* descent (SGD), considers each weight in turn (this can be parallelized)

GRADIENT

- The ratio (difference in loss) / (difference in weight) is approximately the *partial derivative* of the loss function with respect to that weight, i.e.,

$$\frac{\partial C(w, b)}{\partial w_i}$$

Step 3: Find a new weight combination that makes smaller errors

The *optimizer*, usually a variation of stochastic gradient *descent* (SGD), considers each weight in turn (this can be parallelized)

DESCENT

- *Reduce* the weight by an amount proportional to $\frac{\partial C(w,b)}{\partial w_i}$
- Weights that contribute a lot (large derivative) get prioritized
- Compute how much the total error (the loss) would be reduced by if that weight is adjusted by a fixed small amount (the *learning rate*)

Step 3: Find a new weight combination that makes smaller errors

The *optimizer*, usually a variation of *stochastic* gradient descent (SGD), considers each weight in turn (this can be parallelized)

STOCHASTIC

- The partial derivative is only an approximation of the true change in loss
- Calculate based on a *randomly* selected subset of the data

Why not exhaustive search?

Curse of dimensionality

- Think of the example with the lost keys; exhaustive search in six million dimensions is prohibitive

If the loss function is differentiable, the gradient acts like a metal detector

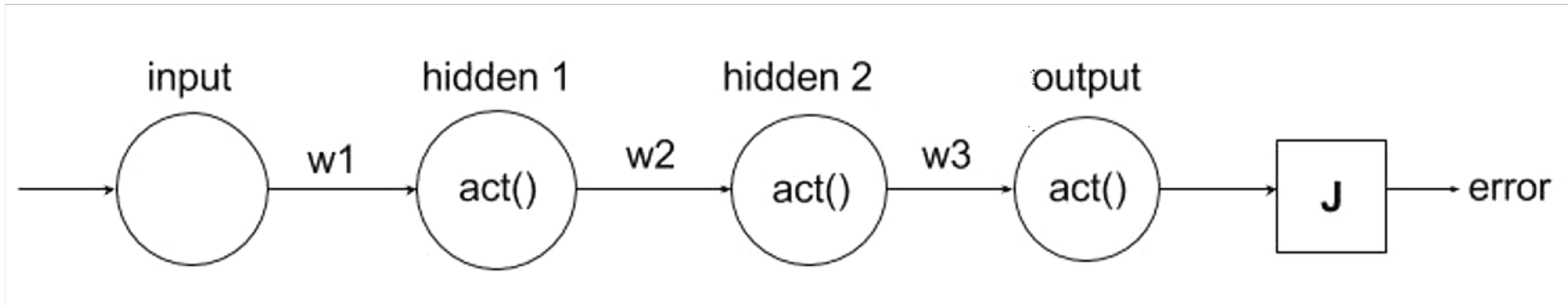
- At any location in the yard, the detector will point you in the direction where the signal gain is strongest

Gradient descent allows you to take incremental steps towards the optimal solution

- Walk a step in the direction of the gradient and recalculate

Need to compute the gradient $\frac{\partial C(w, b)}{\partial w_i}$

Neural network layers are “stacked”:
the input for one is the output of another

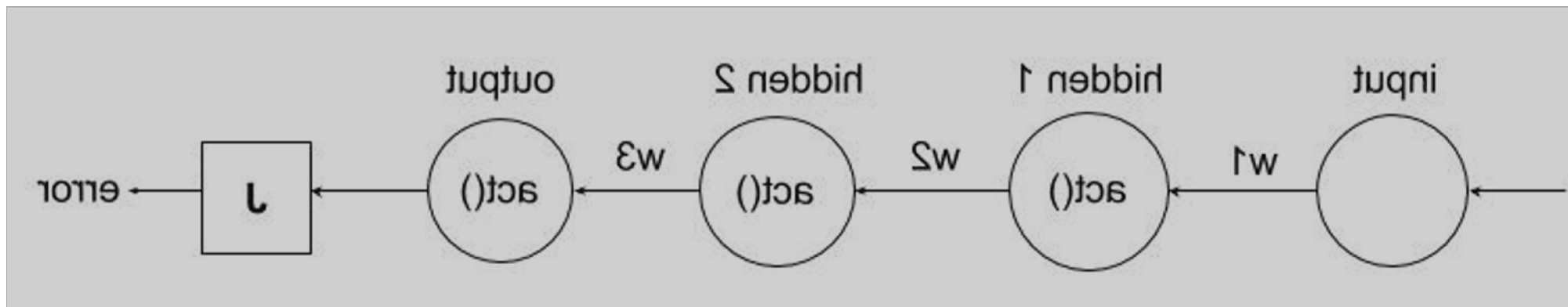


This is a composition of functions

Need to compute the gradient $\frac{\partial C(w, b)}{\partial w_i}$

Neural network layers are “stacked”:

the input for one is the output of another



This is a composition of functions, so we can use the chain rule from calculus; work backwards from the last (top) layer in:

$$\frac{\partial error}{\partial w1} = \frac{\partial error}{\partial output} * \frac{\partial output}{\partial hidden2} * \frac{\partial hidden2}{\partial hidden1} * \frac{\partial hidden1}{\partial w1}$$

Repeat Steps 1-3 until you reach the point of diminishing returns

Each iteration's sample of training data is called a *minibatch* (or often just "batch")

A complete round of the training data is called an *epoch*.

The number of epochs you train for is how many times the network will see each training example.

Batches and epoch

Balance between:

- true stochastic gradient descent (calculate and update separately for each training example)
- true batch gradient descent (calculate and update based on all training examples)

Split the training dataset into small batches of size `batch_size`

Calculate model error and update model coefficients one batch at a time

$$(\text{\# of epochs}) * (\text{batch_size}) = \text{\# of training examples}$$

Other important choices

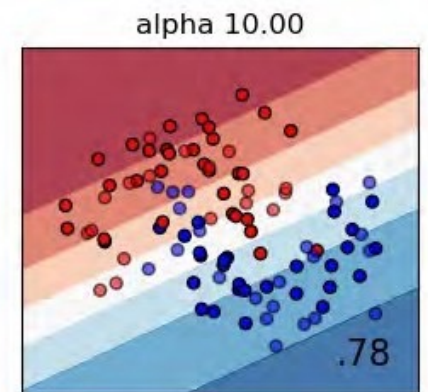
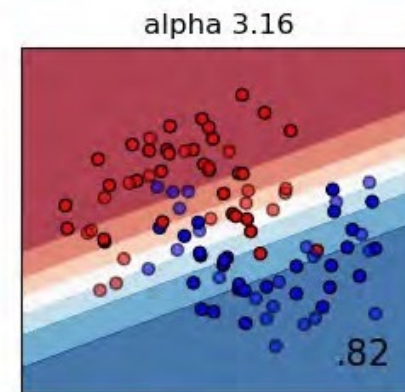
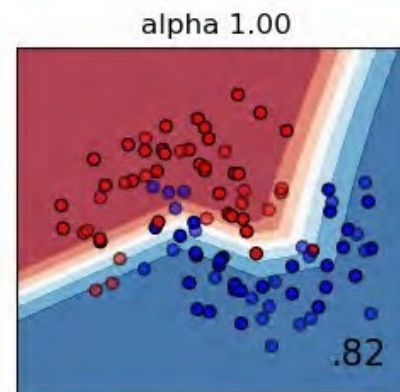
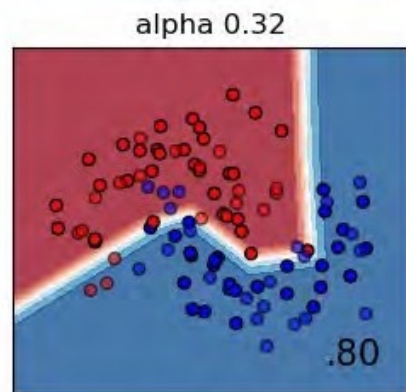
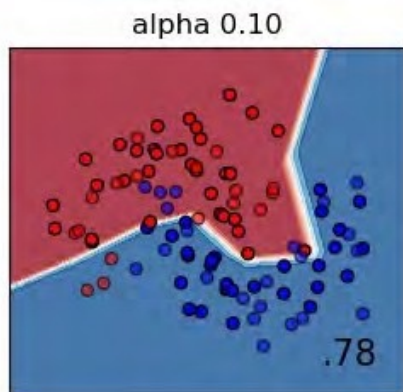
accepting (word
article).
focus n point
converging rays of light,
heat, waves of sound, meet;
centre of activity or
intensity; pl focuses, foci; v
adjust; cause to converge;
concentrate; a focal
pertaining to focus

Alpha

Default is close to zero, so little regularization



Higher alpha means stronger regularization
(less prone to variance/overfitting,
but more prone to bias/underfitting)



Learning rate

The learning rate determines how far to go in the direction that makes the error smaller.

- For example, the optimizer may estimate that a change of one in the value of the weight connecting input variable X_{143} to the 54th ReLU of the first hidden layer will reduce the error by 3.72 units

But this is only an approximation of the true change.

- How big a step to take? How much to change the weight?

Learning rate

The learning rate has a small positive value, often in the range between 0.0 and 1.0

Smaller learning rates mean more conservative changes in weights, taking smaller steps so the approximation will stay valid

- This requires more training epochs; more steps to travel down the optimization path

Larger learning rates can also cause problems; the approximation may no longer hold, and the error is not actually reduced in an optimal way

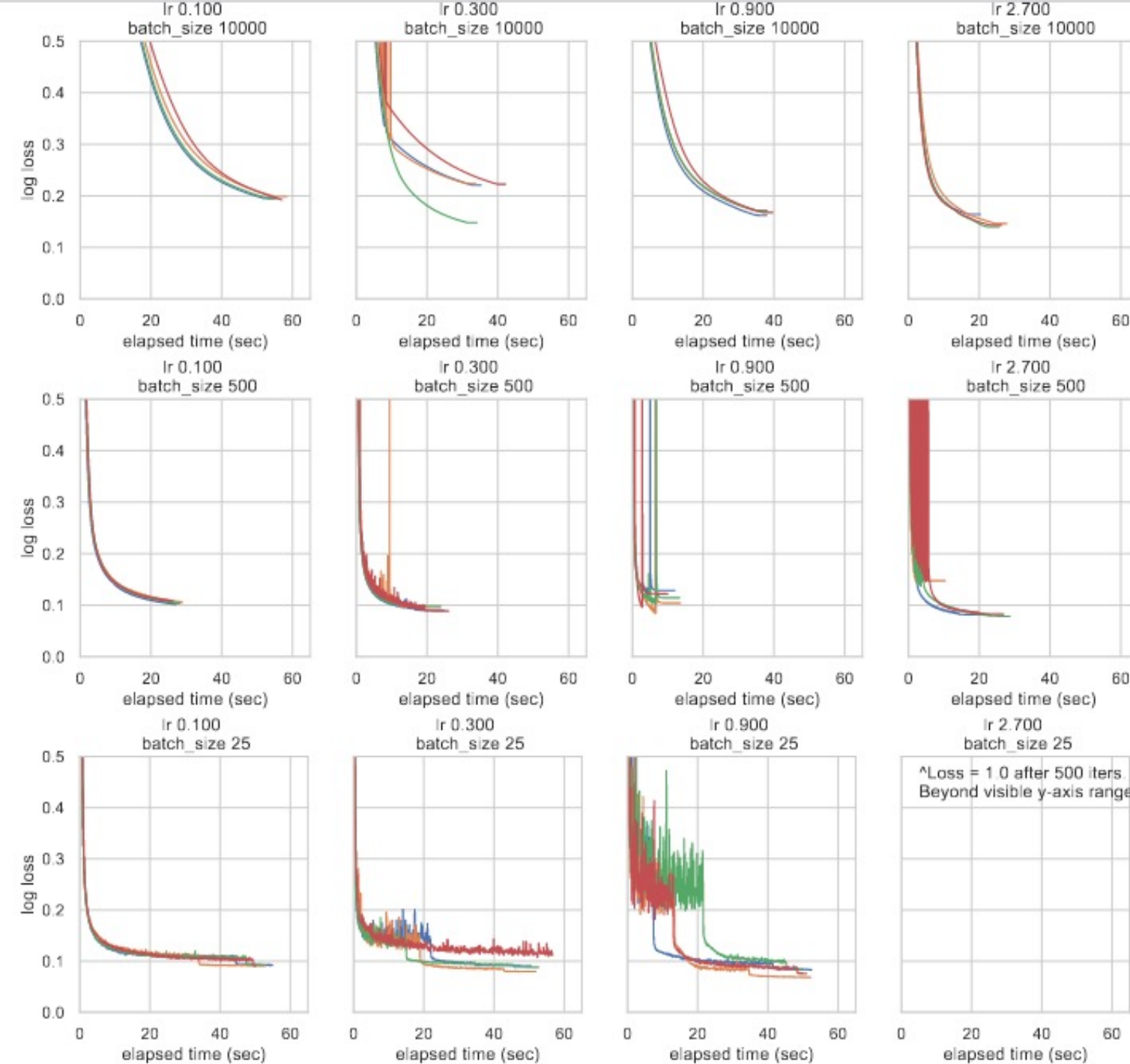
- The model might converge too quickly to a suboptimal solution (rushing to make hasty decisions)

Learning rate

Instead of a constant learning rate, it is recommended to use a high learning rate during the start and reduce it during training.

Typically optimizers take care of this automatically.

Adam is a type of SGD algorithm that has an adaptive learning rate that makes it suitable for most problems without any parameter tuning (it is "self tuning", in a sense); it is a great general-purpose optimizer



**Batch size and
learning rate
work together**

Homework Assignment #3

Due Monday, July 1st, 11:59 pm (Central)