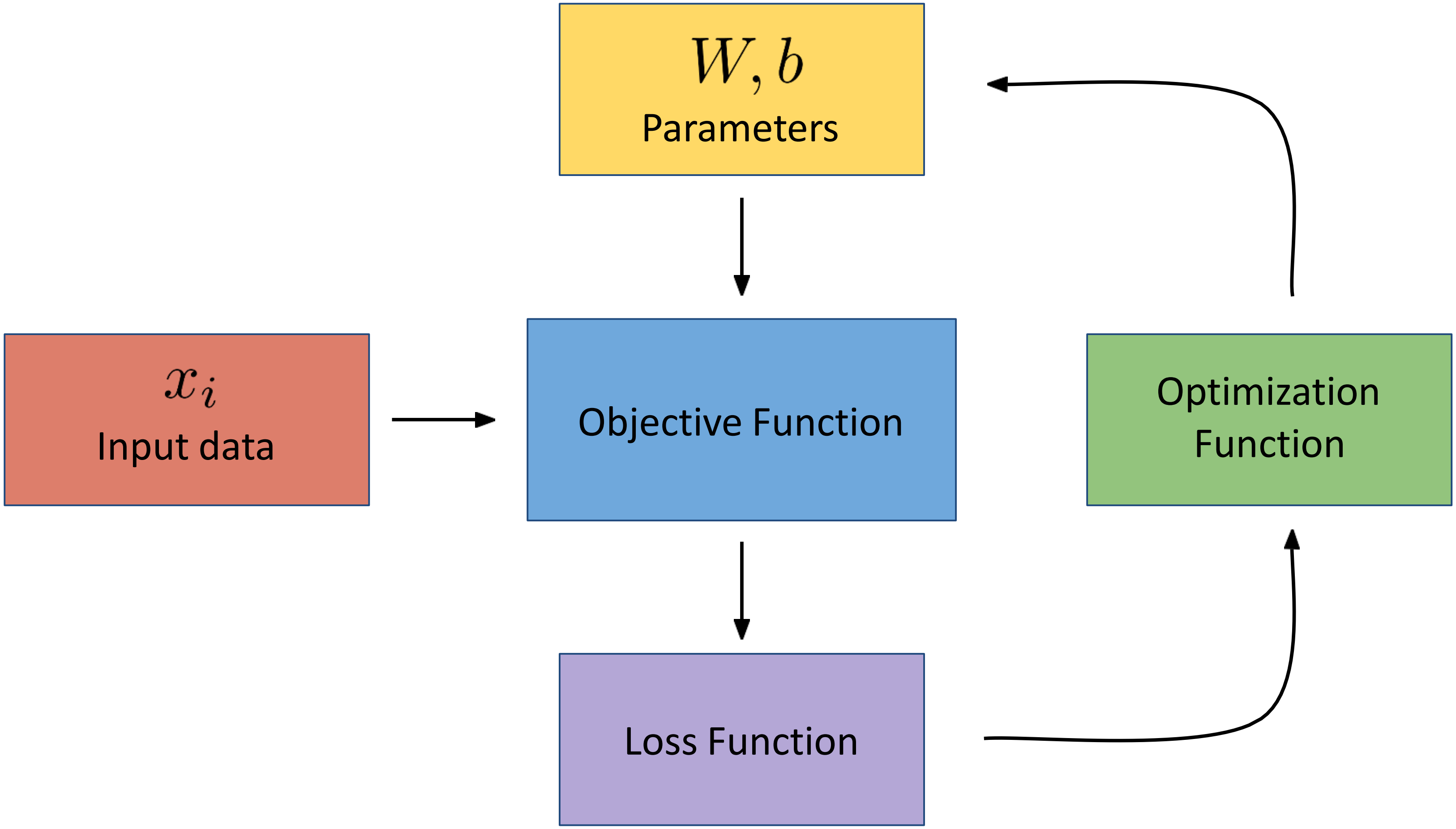
Machine Learning

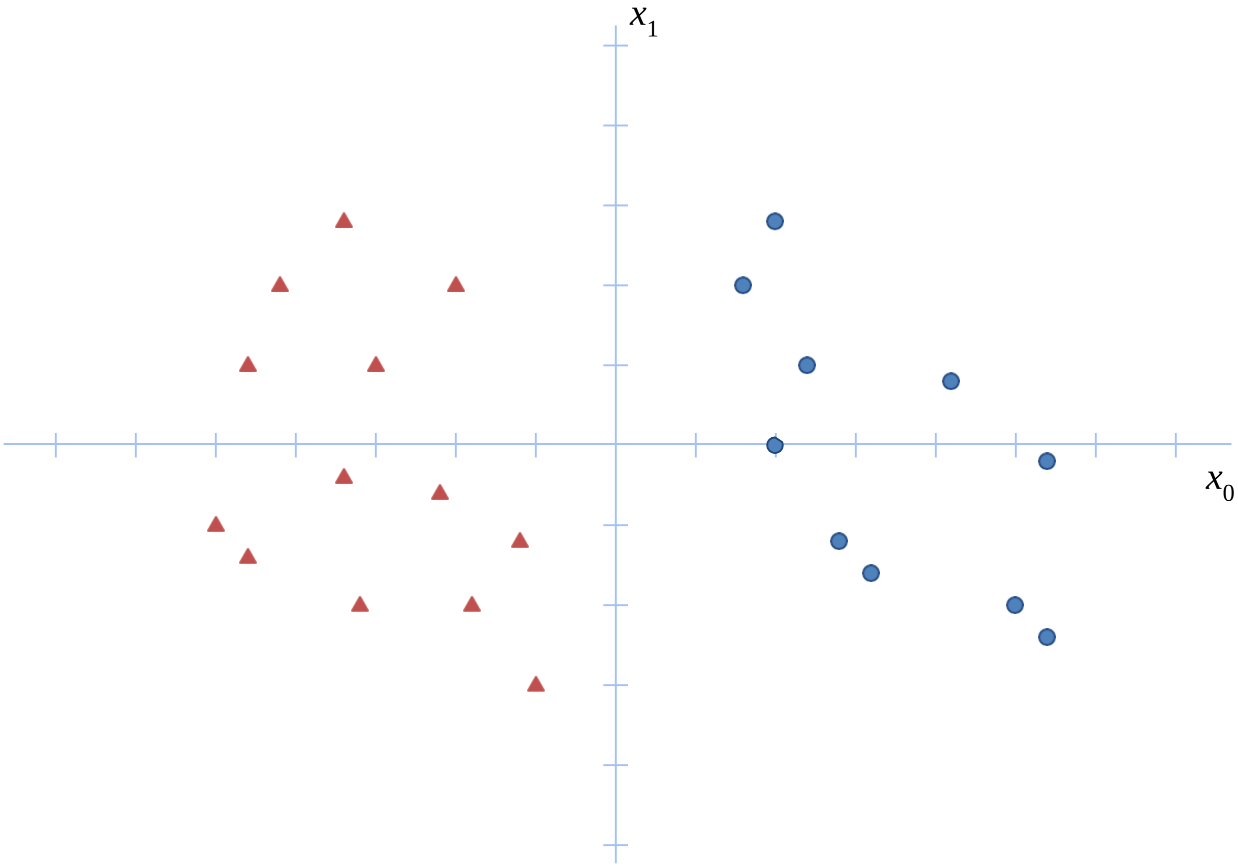
Lecture # 3

Neural Networks

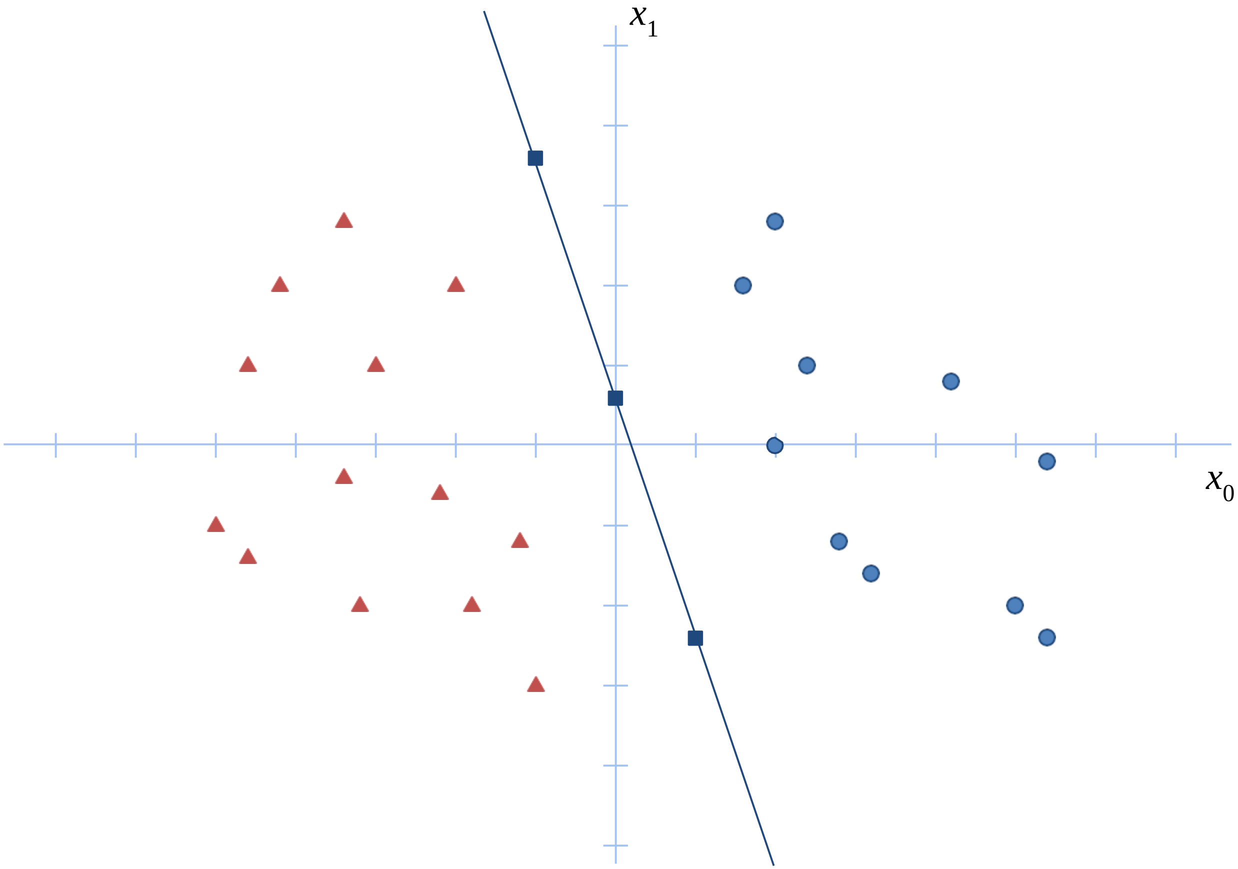
Overall Picture



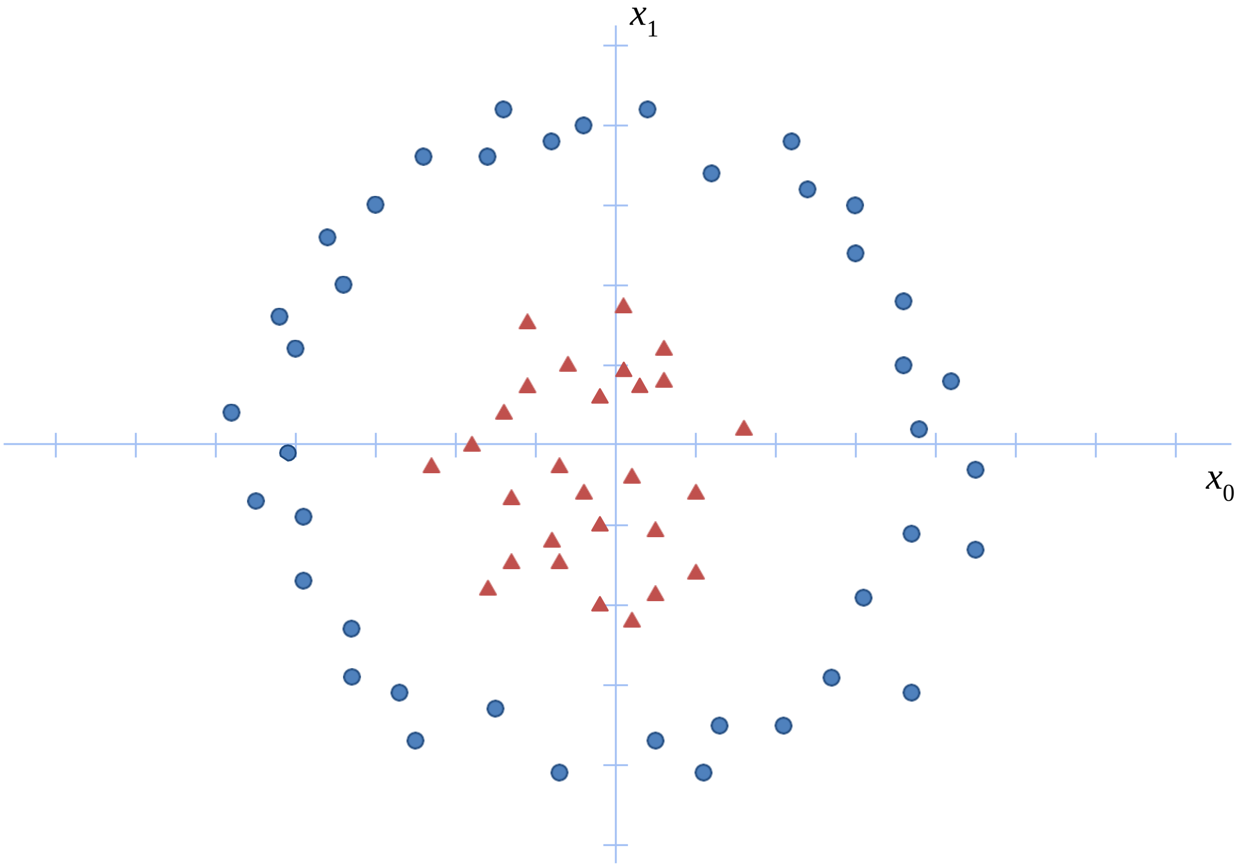
Linear Classifier Recap



Linear Classifier Recap



Linear Classifier?



Linear Separability

Not all problems are linearly classifiable - i.e. if you plot the examples in space, you cannot draw a line/plane to separate them out

Linear Separability

Not all problems are linearly classifiable - i.e. if you plot the examples in space, you cannot draw a line/plane to separate them out

Neural Networks are *one* way to solve this problem

Linear Classifier

$$f(x, W, b)$$

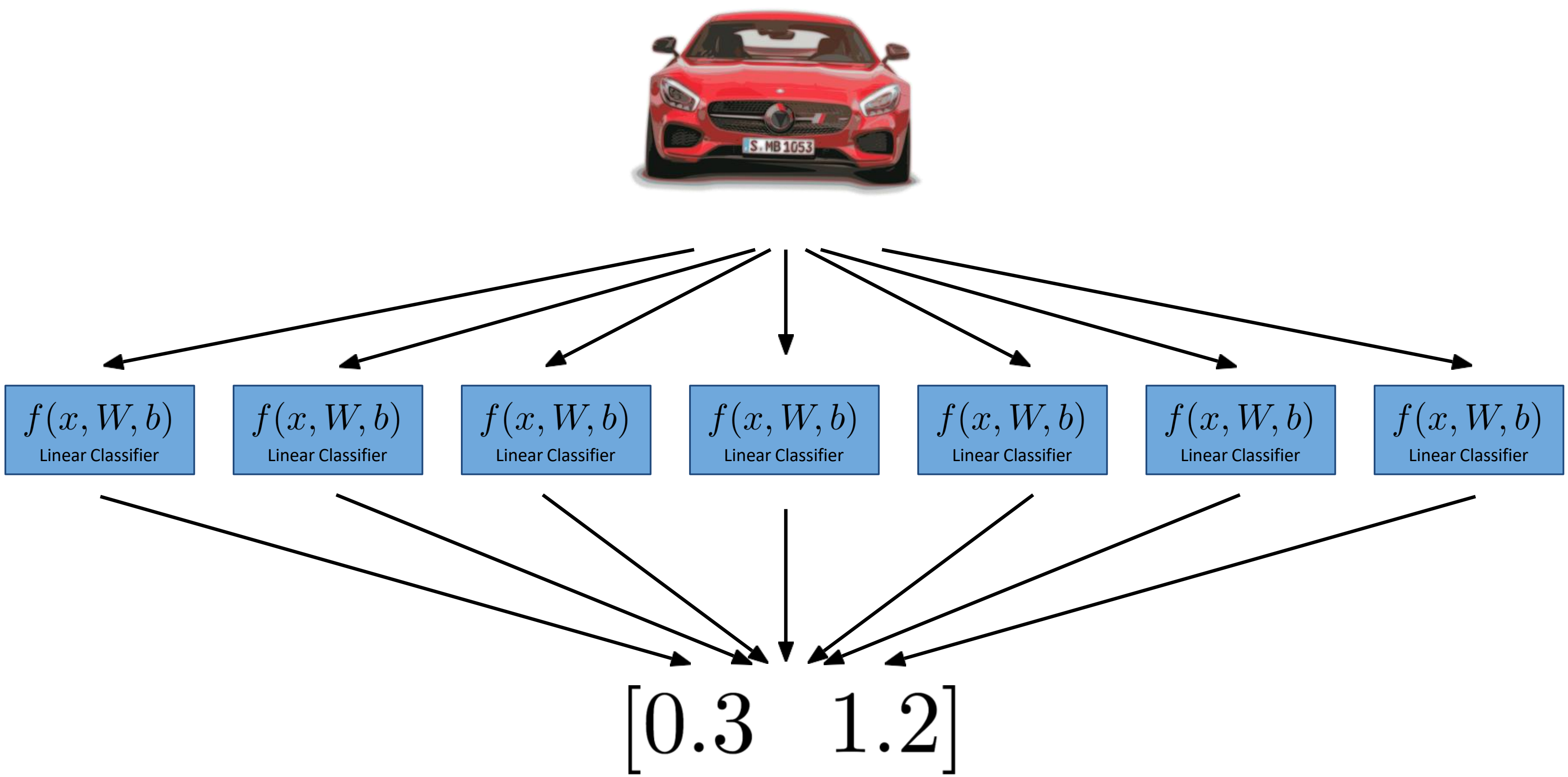
Linear Classifier

$$[0.3 \quad 1.2]$$

Score for the car to be accident prone

Score for the car to be *NOT* accident prone

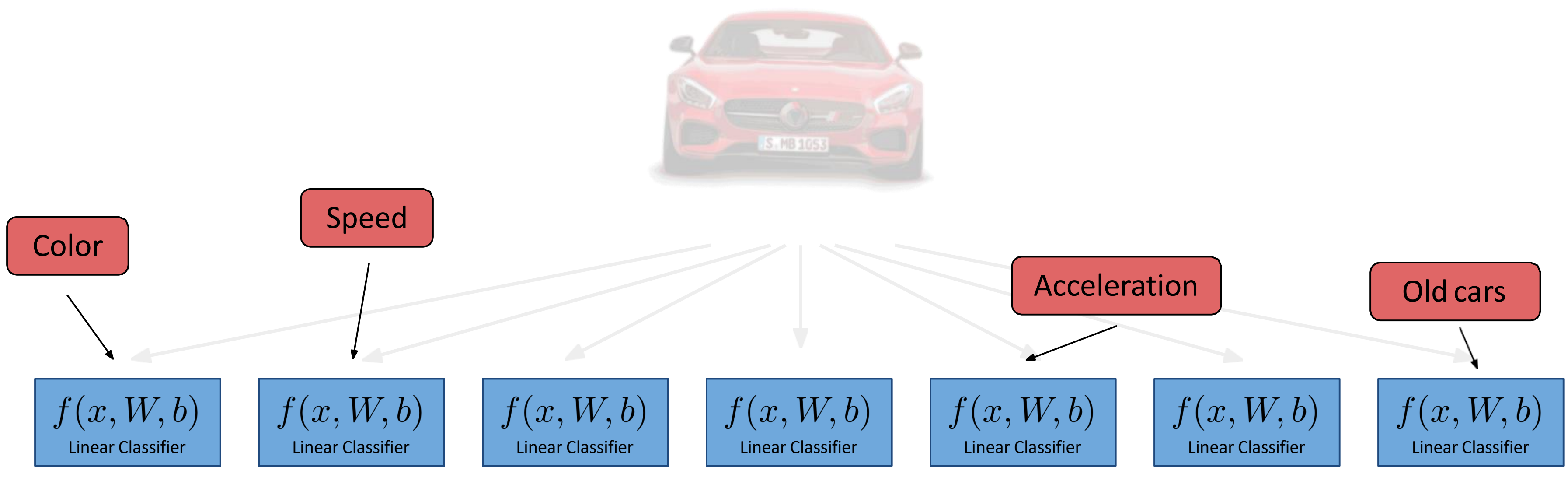
Neural Network



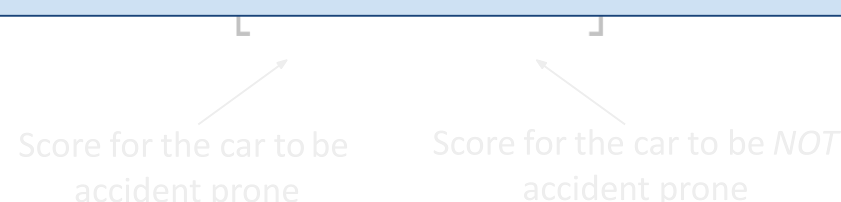
Score for the car to be accident prone

Score for the car to be *NOT* accident prone

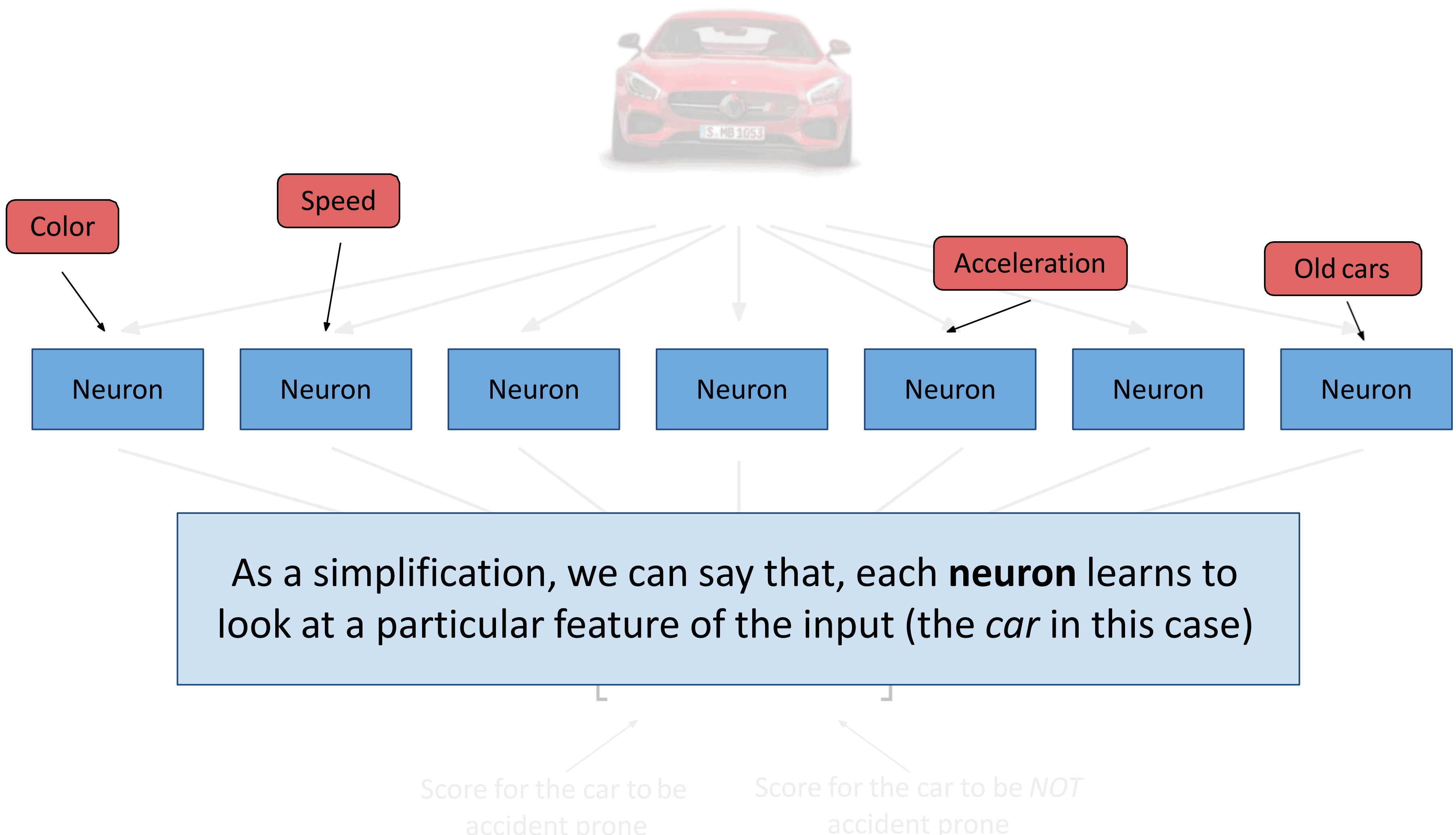
Neural Network



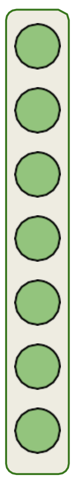
As a simplification, we can say that, each classifier learns to look at a particular feature of the input (the *car* in this case)



Neural Network

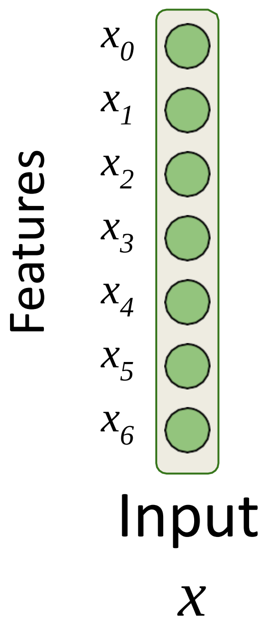


Neural Network

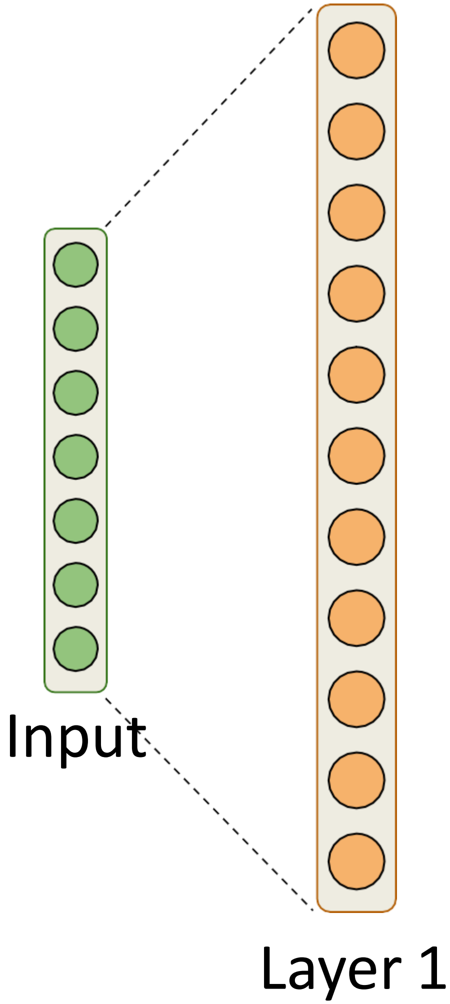


x

Neural Network

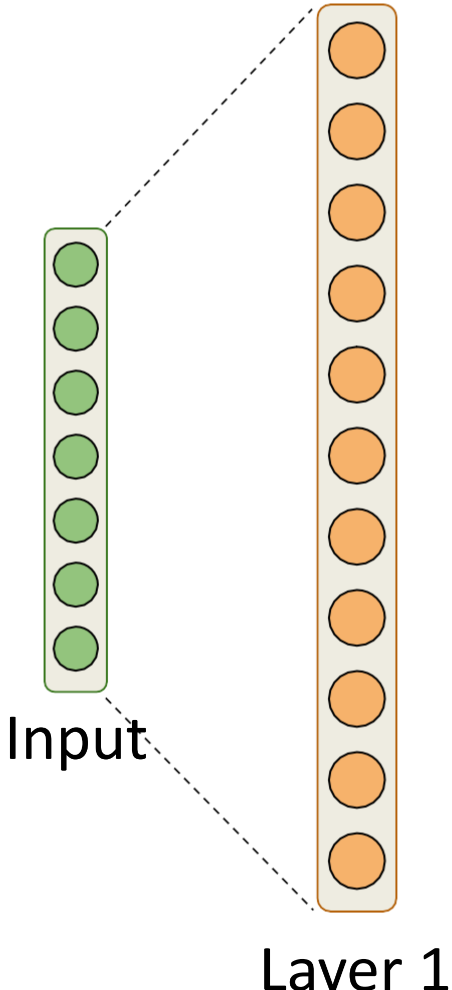


Neural Network



The neurons in the layer can be thought of as representing *richer features*

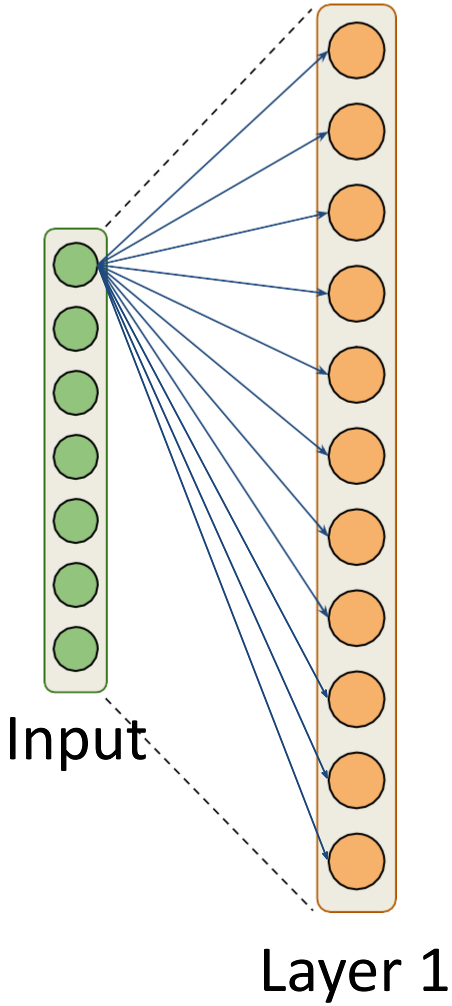
Neural Network



The neurons in the layer can be thought of as representing *richer features*

Think of these *richer features* as combinations of the *input features* we provided to the system

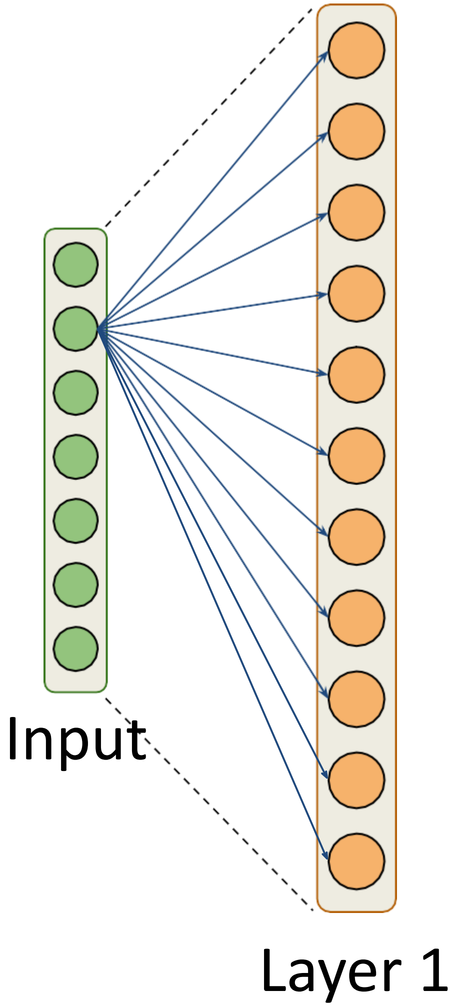
Neural Network



The neurons in the layer can be thought of as representing *richer features*

Think of these *richer features* as combinations of the *input features* we provided to the system

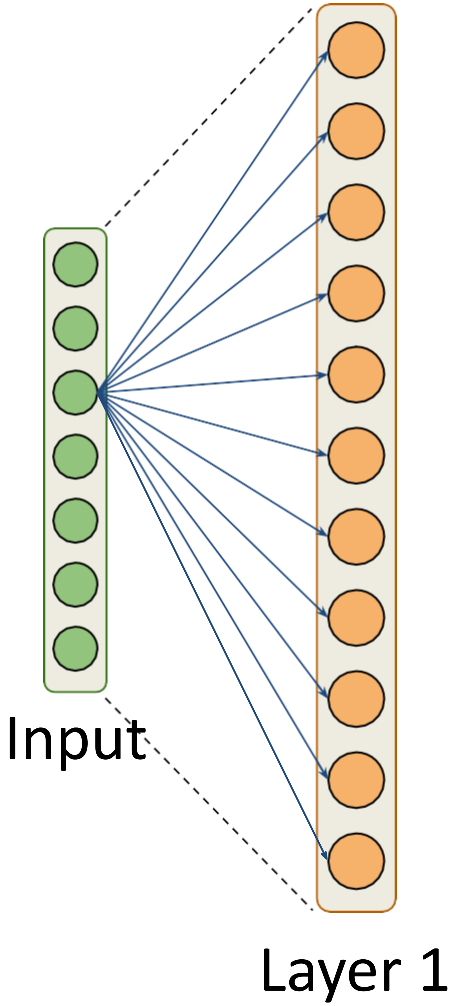
Neural Network



The neurons in the layer can be thought of as representing *richer features*

Think of these *richer features* as combinations of the *input features* we provided to the system

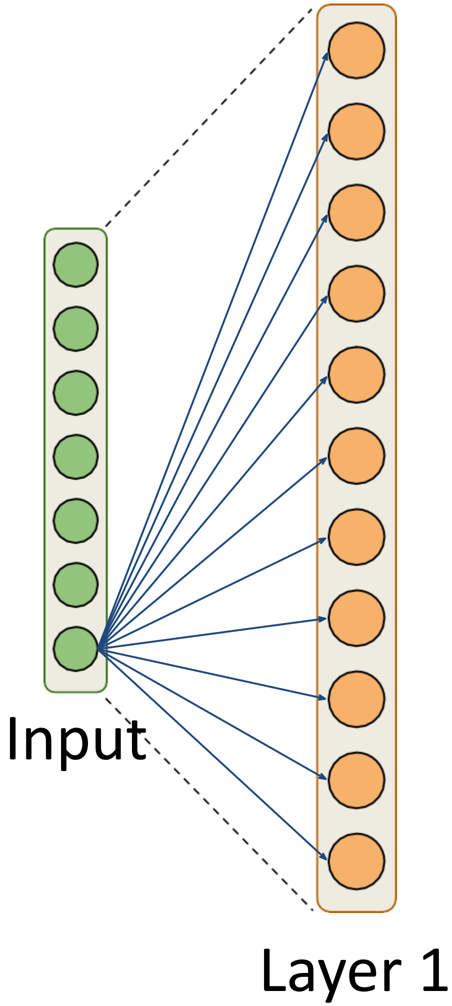
Neural Network



The neurons in the layer can be thought of as representing *richer features*

Think of these *richer features* as combinations of the *input features* we provided to the system

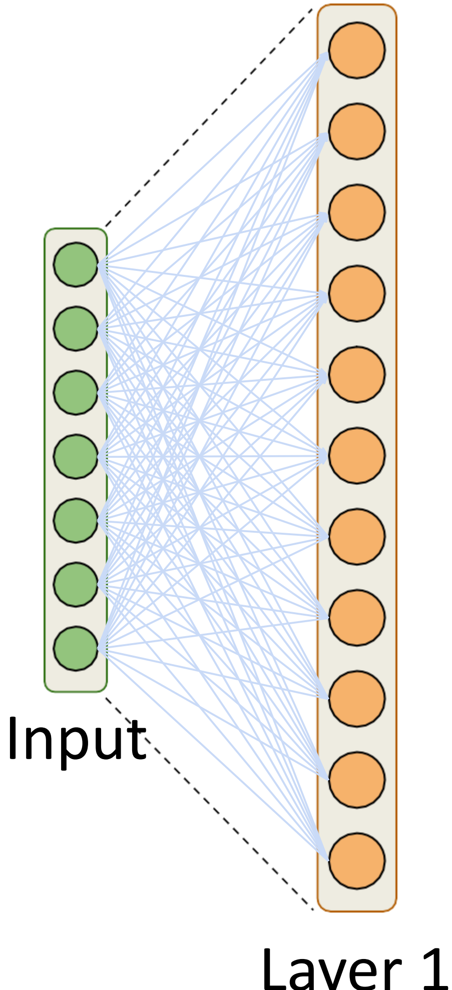
Neural Network



The neurons in the layer can be thought of as representing *richer features*

Think of these *richer features* as combinations of the *input features* we provided to the system

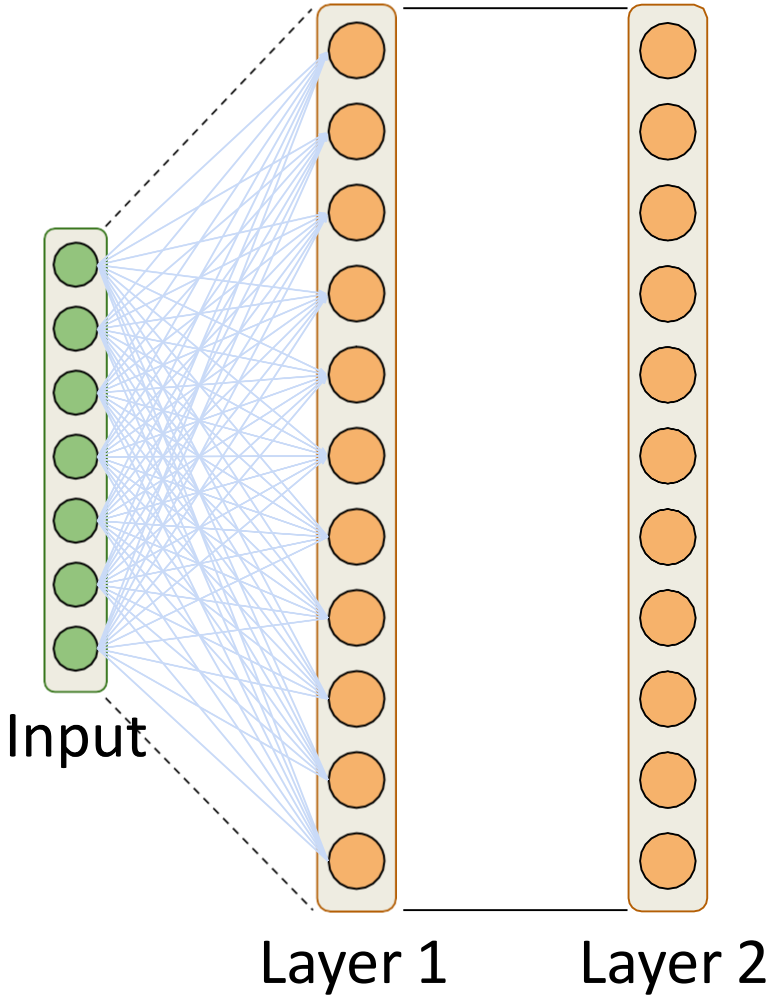
Neural Network



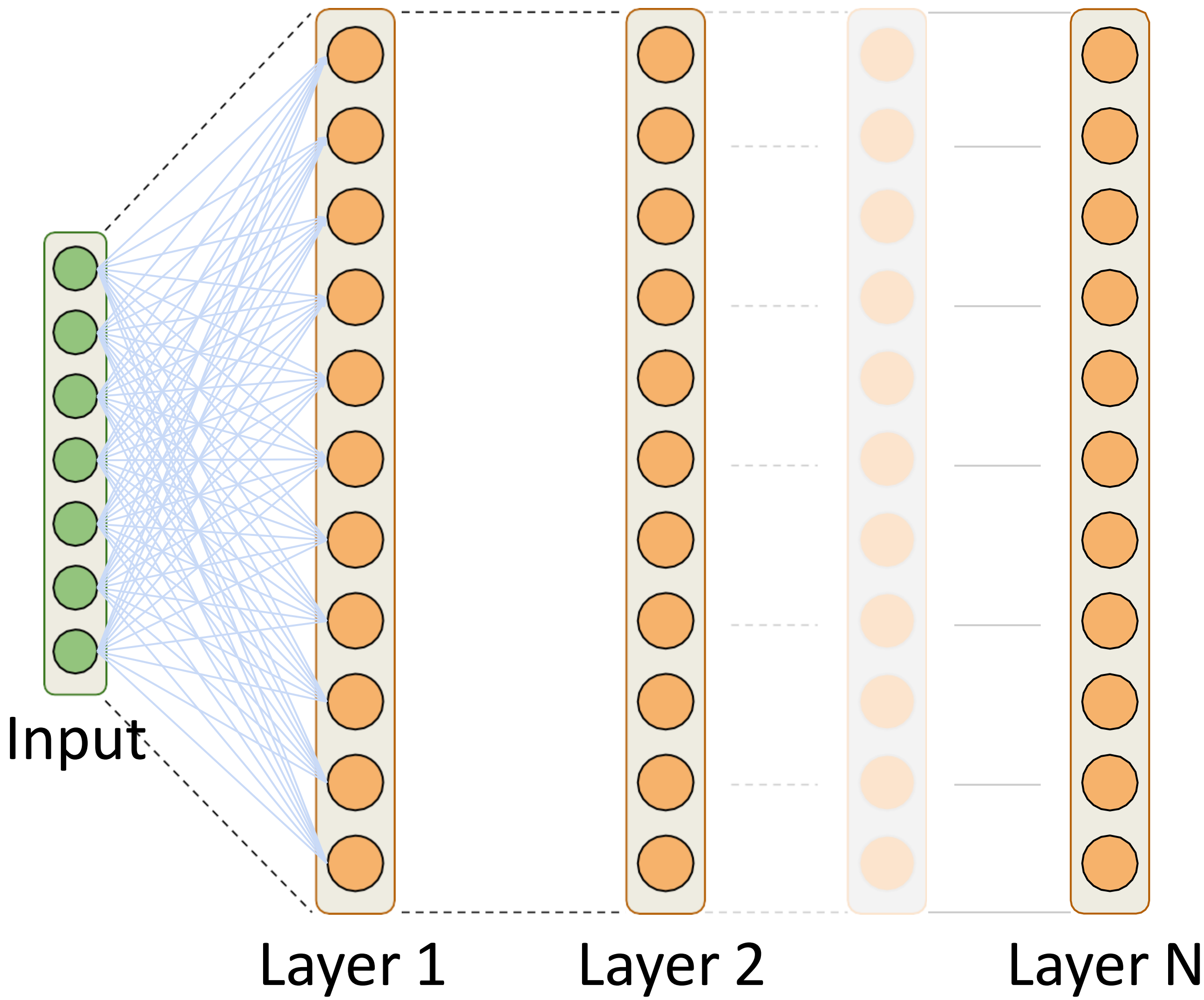
The neurons in the layer can be thought of as representing *richer features*

Think of these *richer features* as combinations of the *input features* we provided to the system

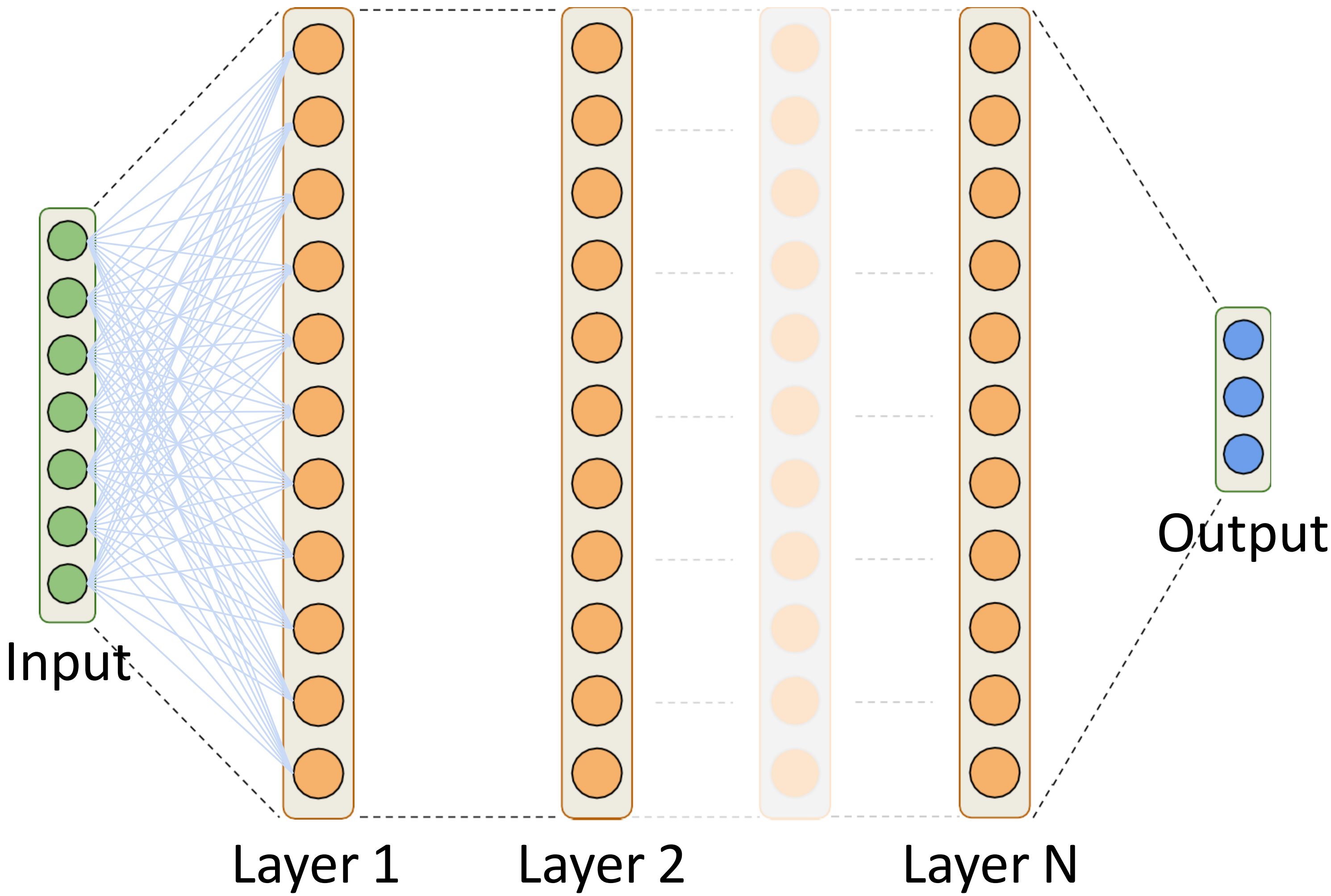
Neural Network



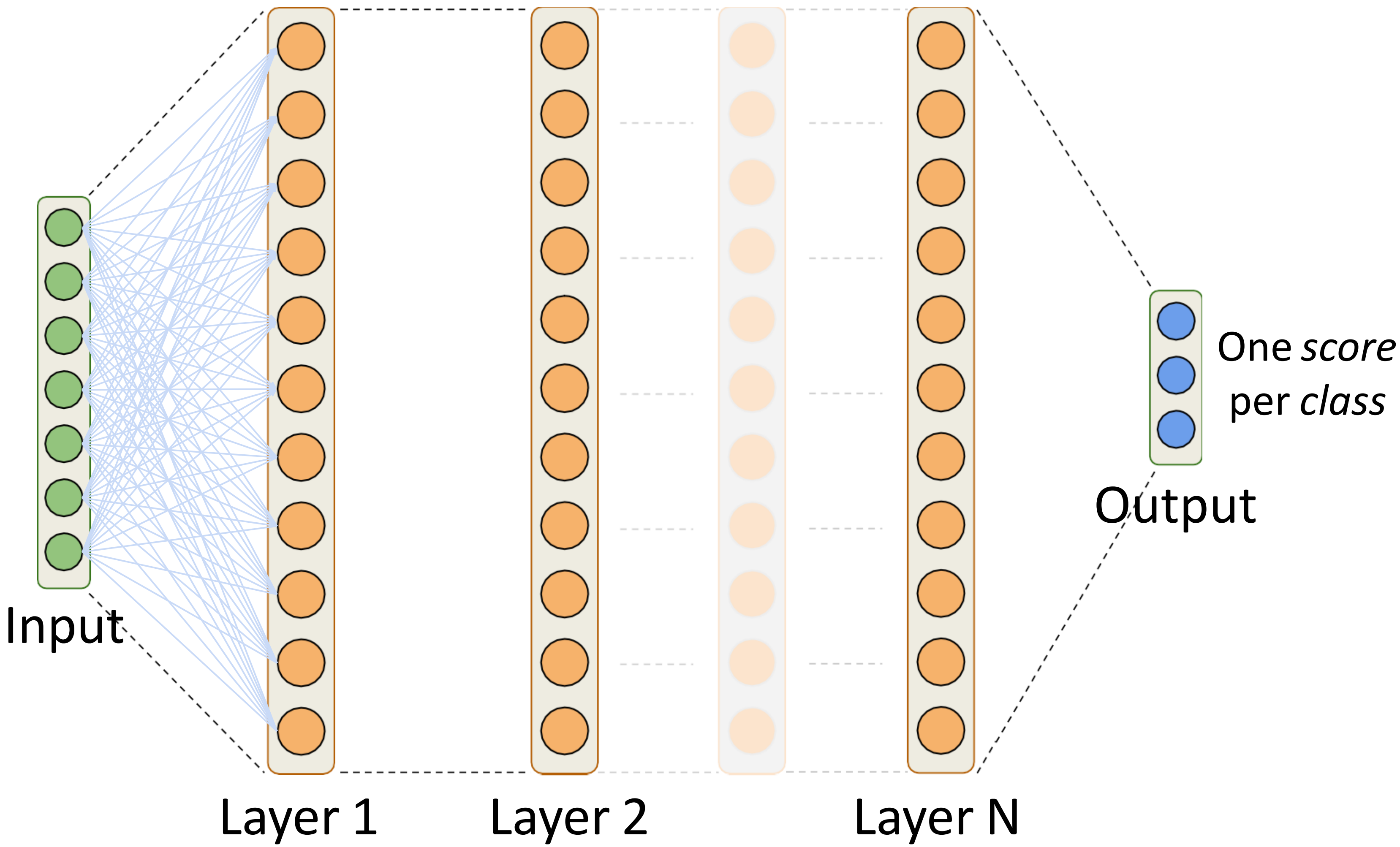
Neural Network



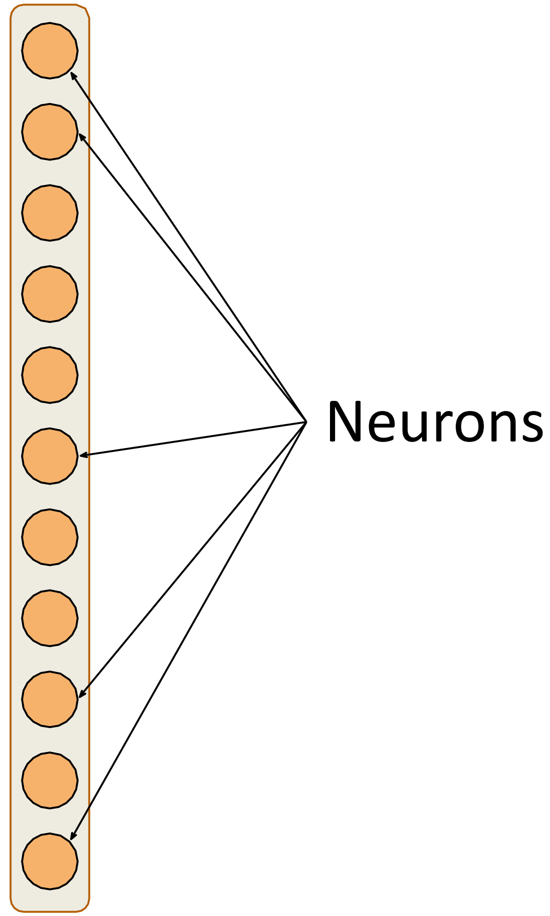
Neural Network



Neural Network

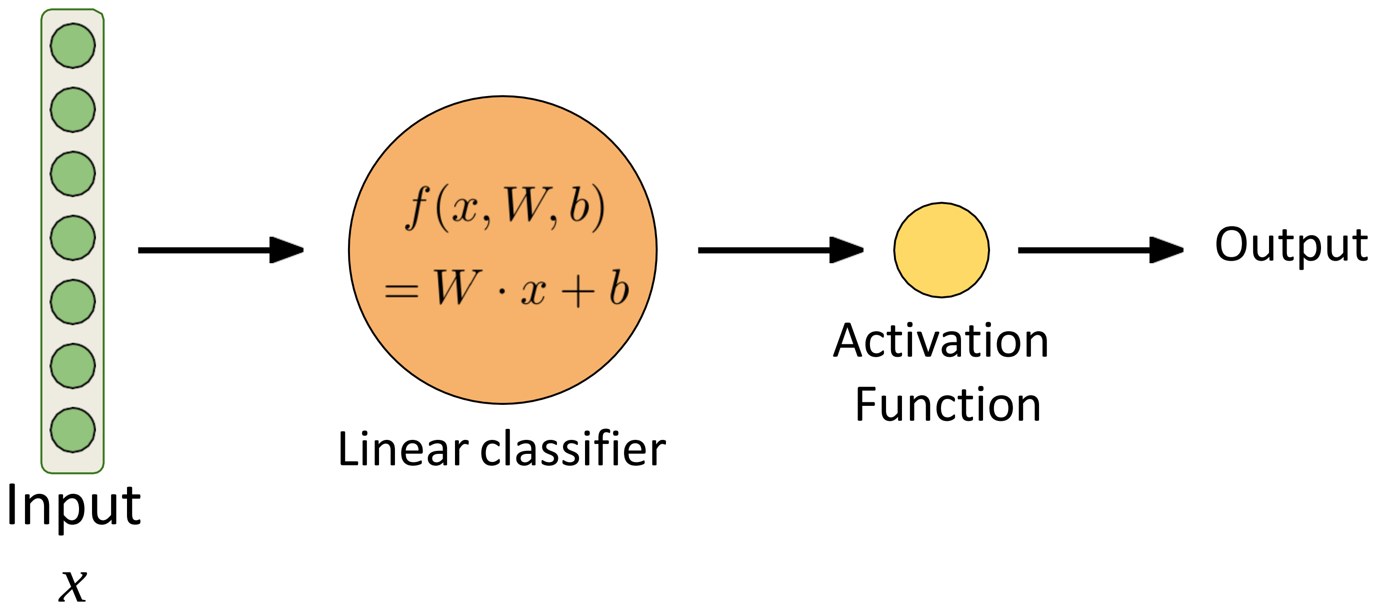


Neural Network



Neuron

A Neuron can be thought of as *a linear classifier* plus an *activation function*



Activation Functions

- Intuitively, a neuron looks at a particular feature of the data

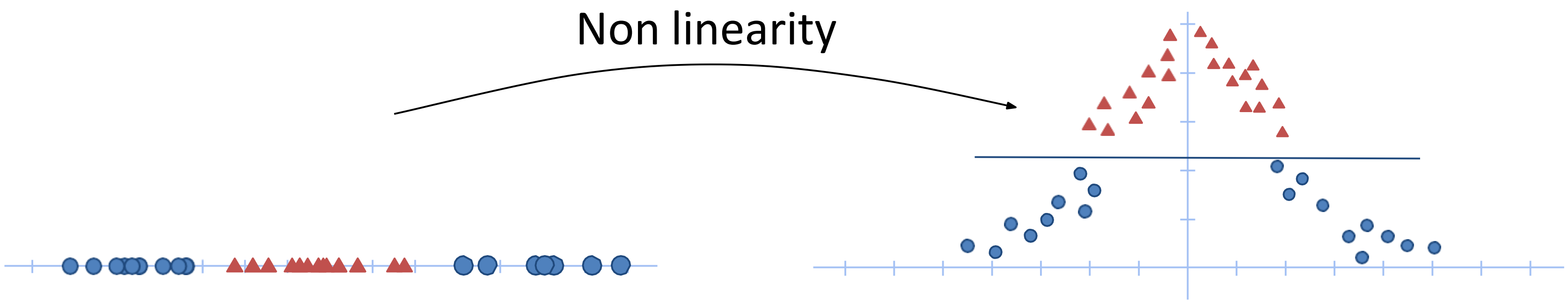
Activation Functions

- Intuitively, a neuron looks at a particular feature of the data
- The activation after the linear classifier gives us an idea of how much the neuron "supports" the feature

As an example, the output of a neuron will be high if the feature it supports is contained in the input
(like "low speed" in the current "car")

Activation Functions

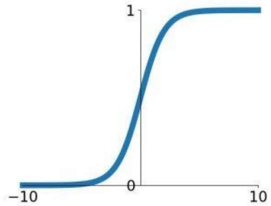
- Intuitively, a neuron looks at a particular feature of the data
- The activation after the linear classifier gives us an idea of how much the neuron “supports” the feature
- Activations also helps us map linear spaces into non-linear spaces



Activation Functions

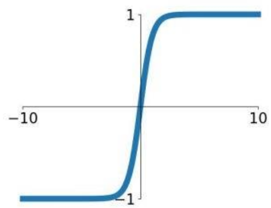
Sigmoid

$\sigma(x) = \frac{1}{1+e^{-x}}$



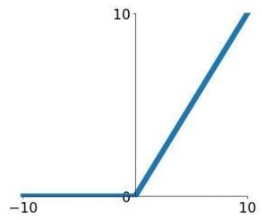
tanh

$\tanh(x)$



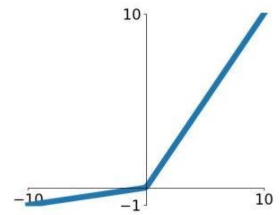
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

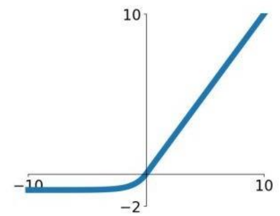


Maxout

$\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Neural Network

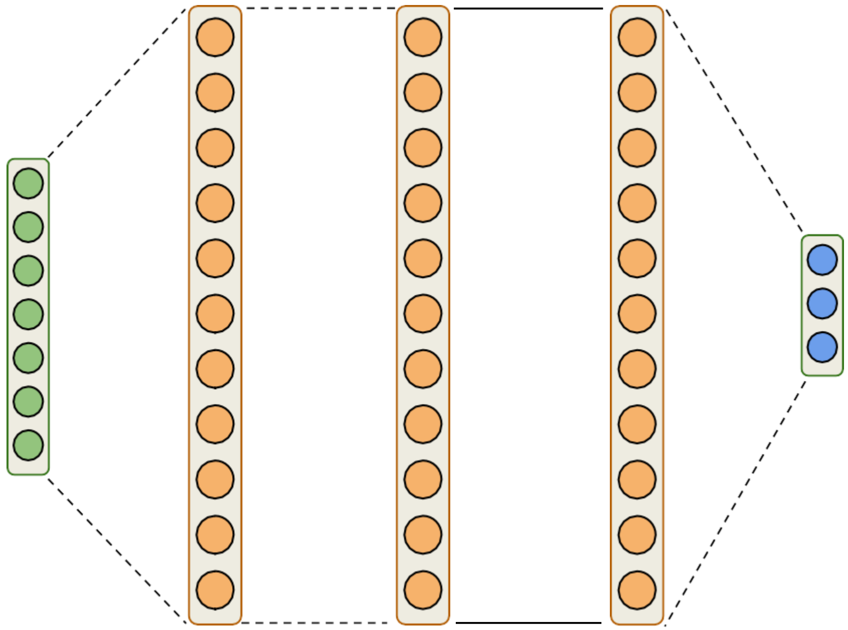
- Entire network is nothing but a function:

$$\boxed{\begin{array}{c} f = W \cdot x + b \\ \text{Linear classifier} \end{array}}$$

Neural Network

- Entire network is nothing but a function:

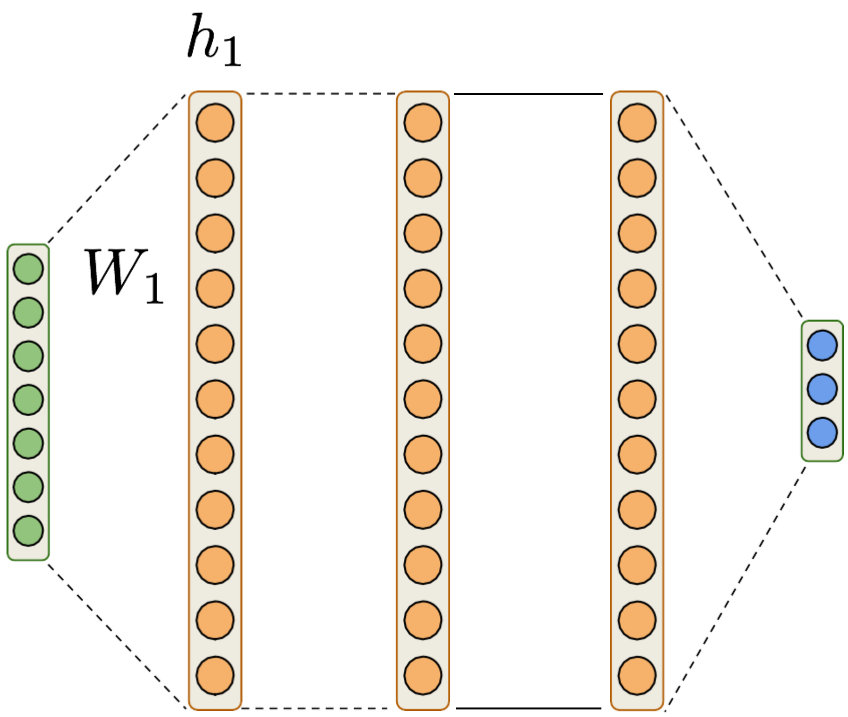
Neural network with 3 hidden layers



Neural Network

- Entire network is nothing but a function:

Neural network with 3 hidden layers

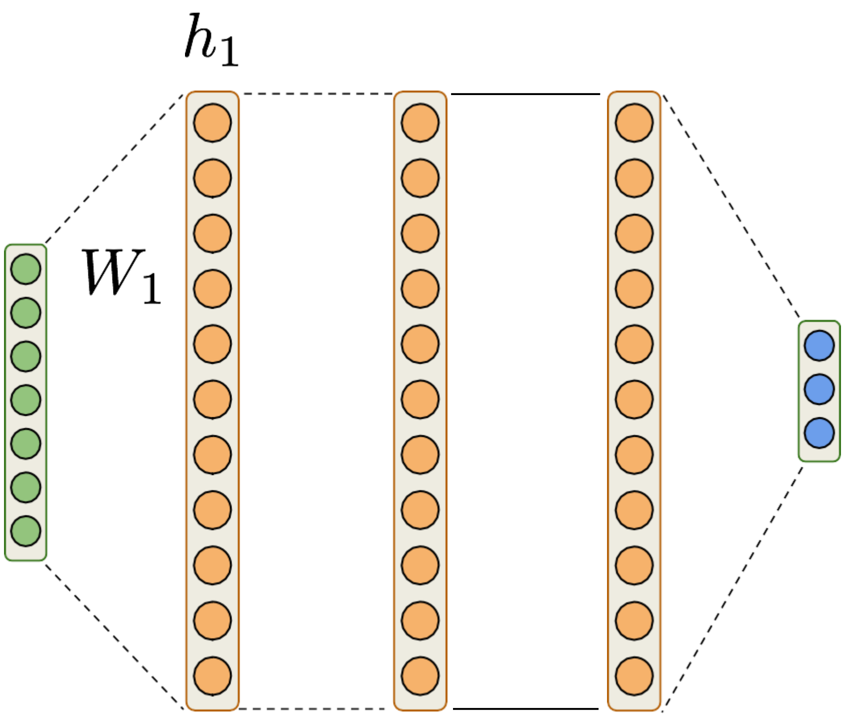


$$h_1 = \sigma(W_1 \cdot x + b_1)$$

Neural Network

- Entire network is nothing but a function:

Neural network with 3 hidden layers



Activation function

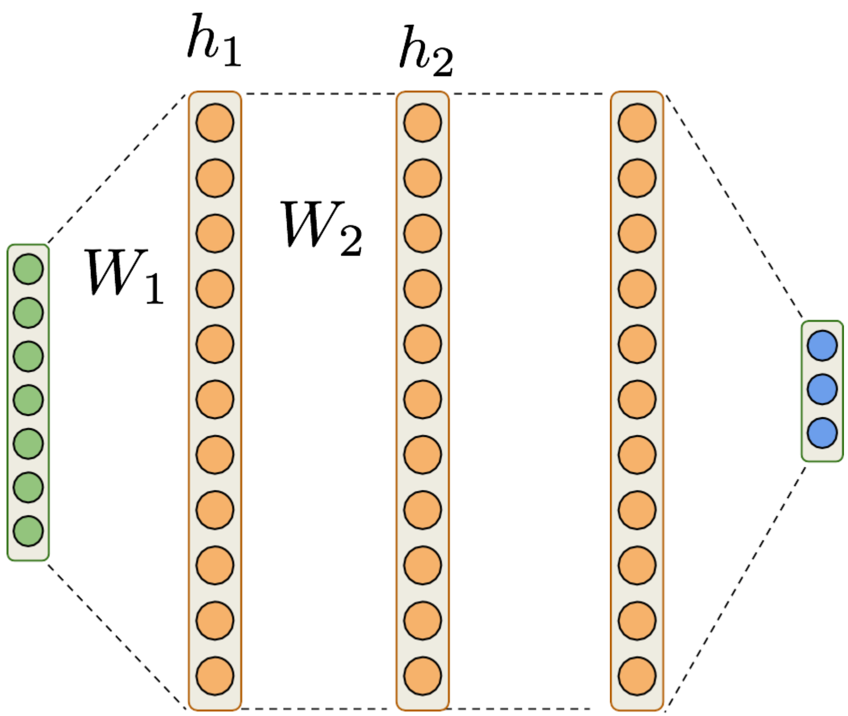
$$h_1 = \boxed{\sigma}(\boxed{W_1 \cdot x + b_1})$$

Output of linear classifier
“richer features”

Neural Network

- Entire network is nothing but a function:

Neural network with 3 hidden layers



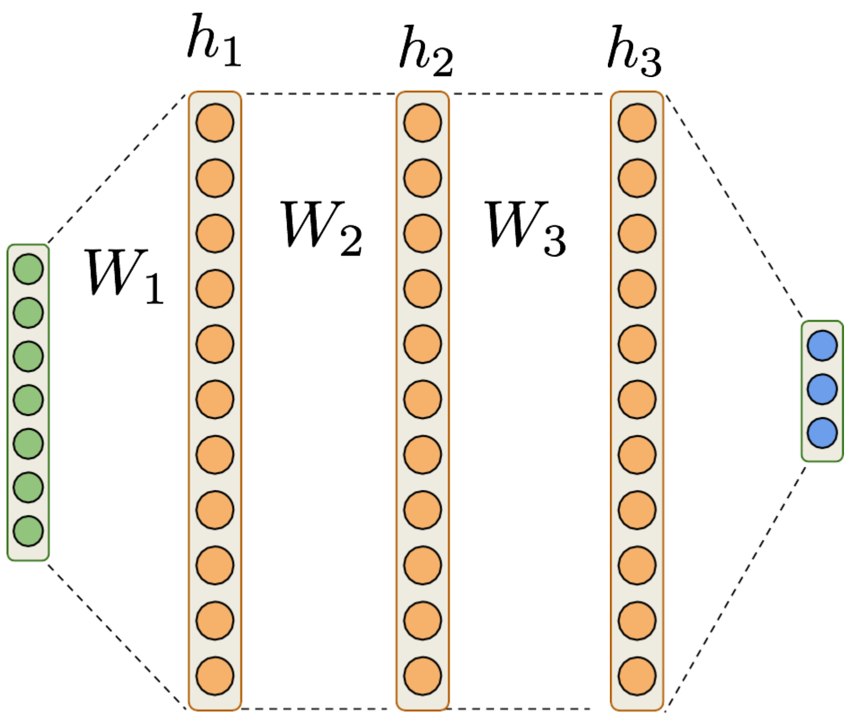
$$h_1 = \sigma(W_1 \cdot x + b_1)$$

$$h_2 = \sigma(W_2 \cdot h_1 + b_2)$$

Neural Network

- Entire network is nothing but a function:

Neural network with 3 hidden layers

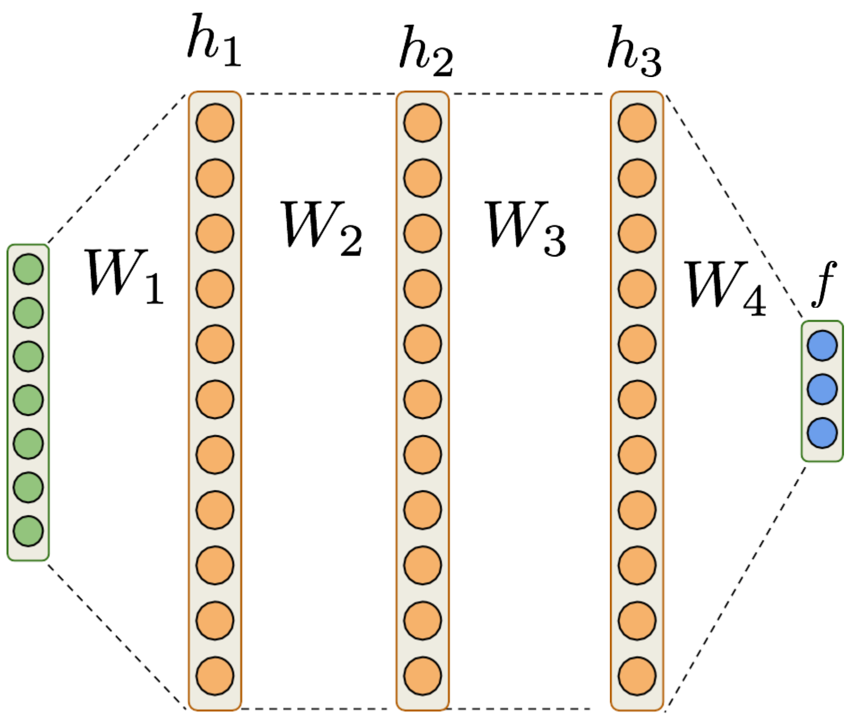


$$h_1 = \sigma(W_1 \cdot x + b_1)$$
$$h_2 = \sigma(W_2 \cdot h_1 + b_2)$$
$$h_3 = \sigma(W_3 \cdot h_2 + b_3)$$

Neural Network

- Entire network is nothing but a function:

Neural network with 3 hidden layers



$$h_1 = \sigma(W_1 \cdot x + b_1)$$

$$h_2 = \sigma(W_2 \cdot h_1 + b_2)$$

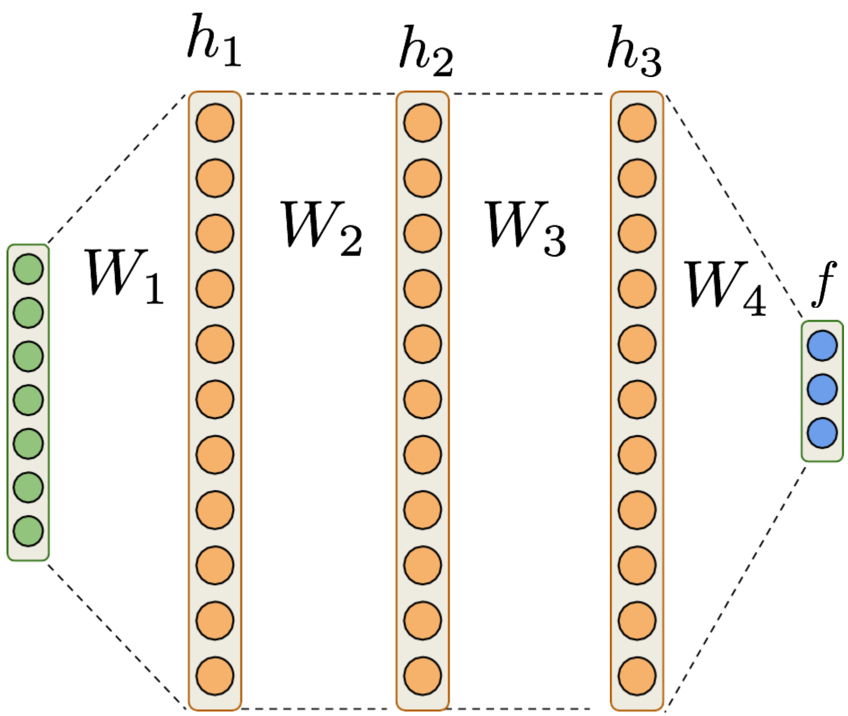
$$h_3 = \sigma(W_3 \cdot h_2 + b_3)$$

$$f = W_4 \cdot h_3 + b_4$$

Neural Network

- Entire network is nothing but a function:

Neural network with 3 hidden layers



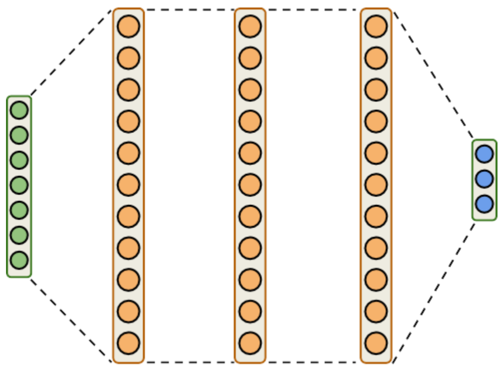
$$h_1 = \sigma(W_1 \cdot x + b_1)$$
$$h_2 = \sigma(W_2 \cdot h_1 + b_2)$$
$$h_3 = \sigma(W_3 \cdot h_2 + b_3)$$
$$f = \underbrace{\boxed{W_4 \cdot h_3 + b_4}}_{\text{Final scores}}$$

Neural Network

- Everything else remains the same!

$$f = W \cdot x + b$$

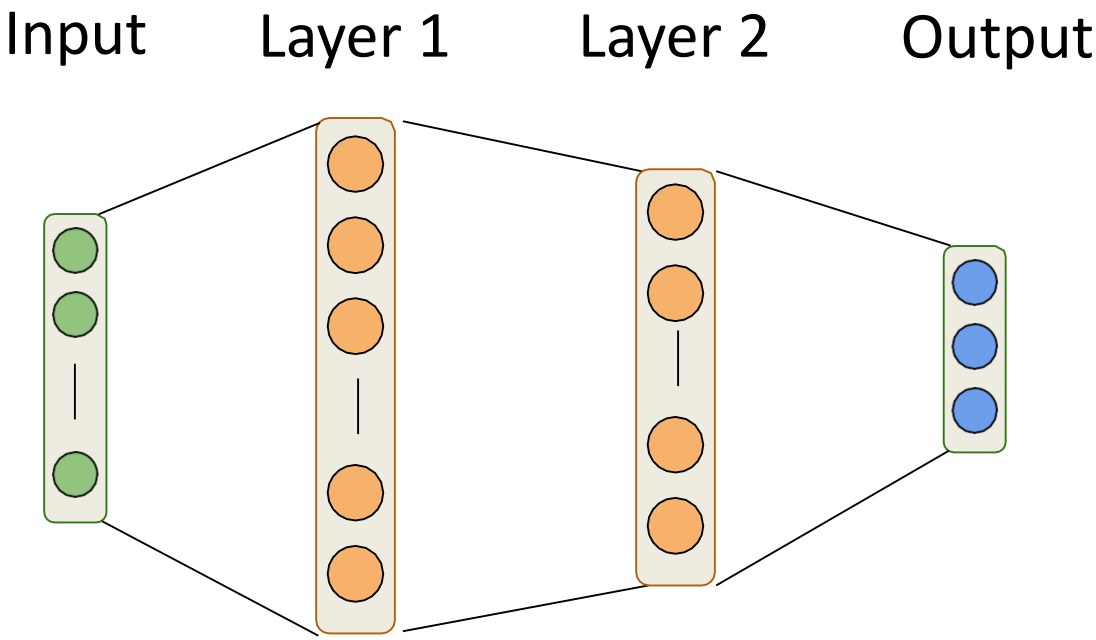
Linear classifier



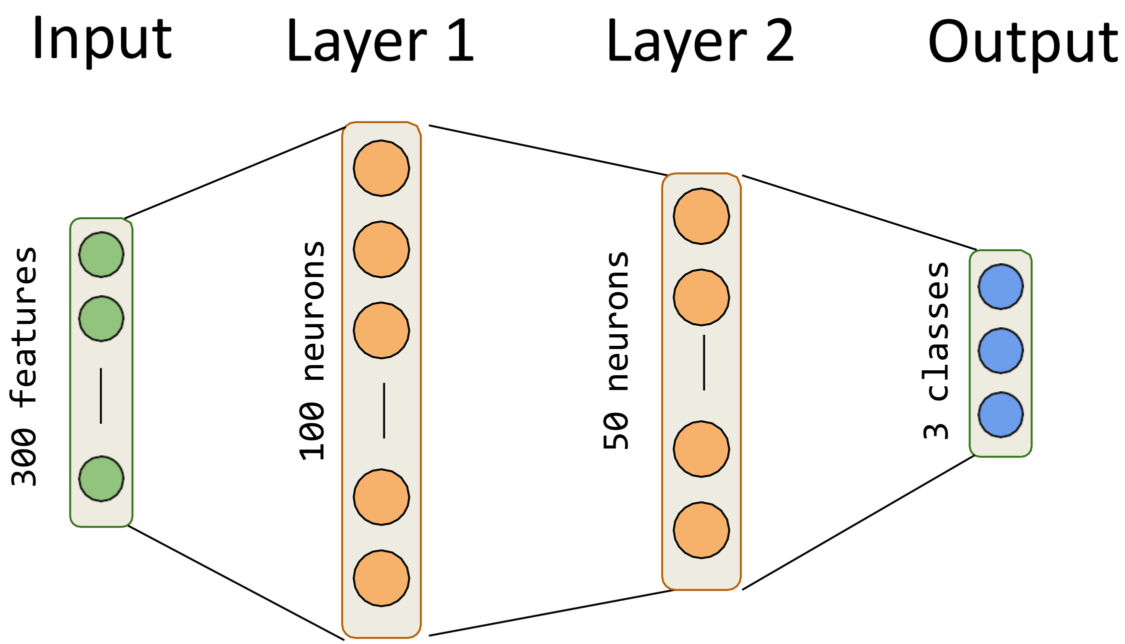
$$f = W_4 \cdot (\sigma(W_3 \cdot (\sigma(W_2 \cdot (\sigma(W_1 \cdot x + b_1)) + b_2)) + b_3)) + b_4$$

Neural network with 3 hidden layers

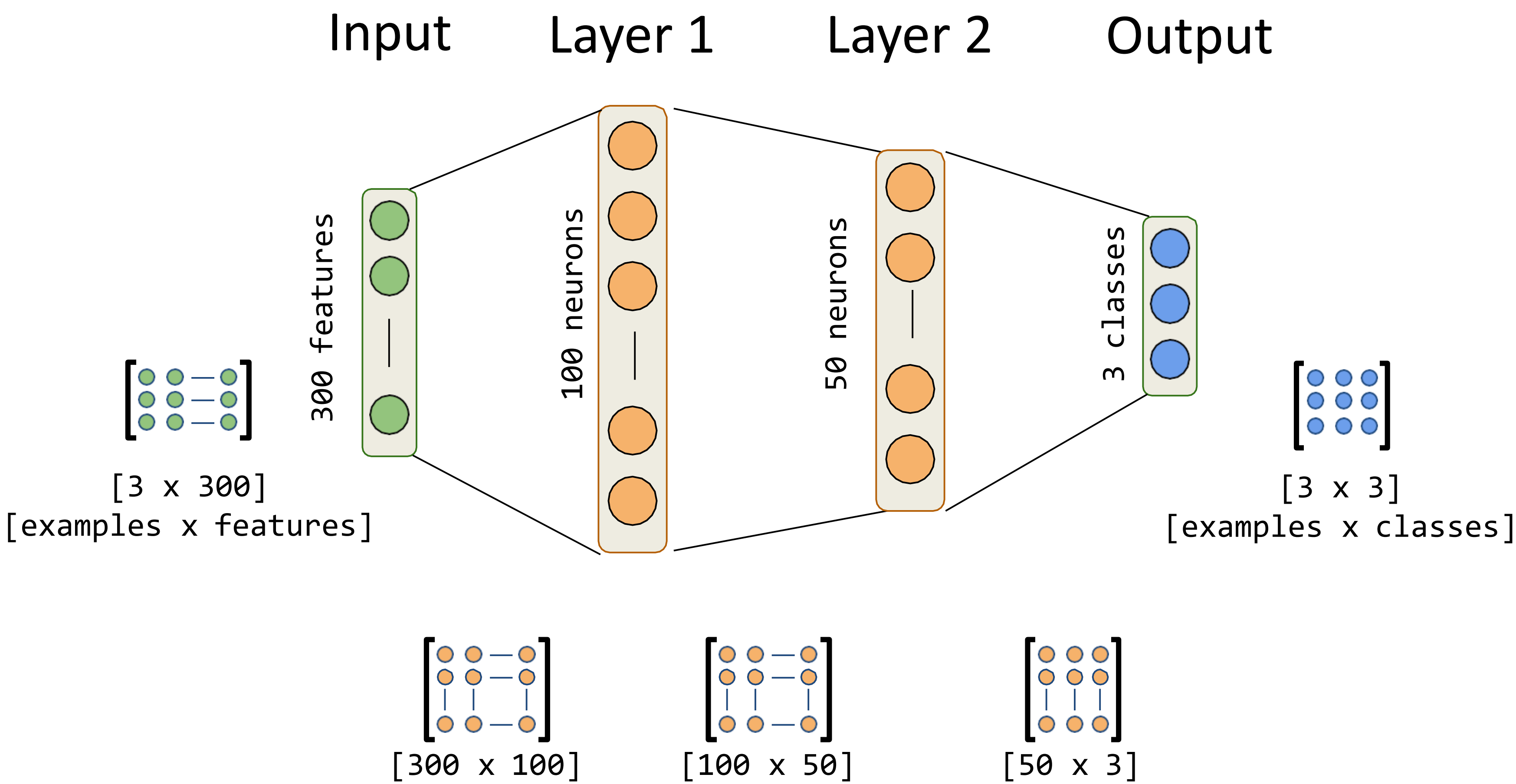
Neural Network



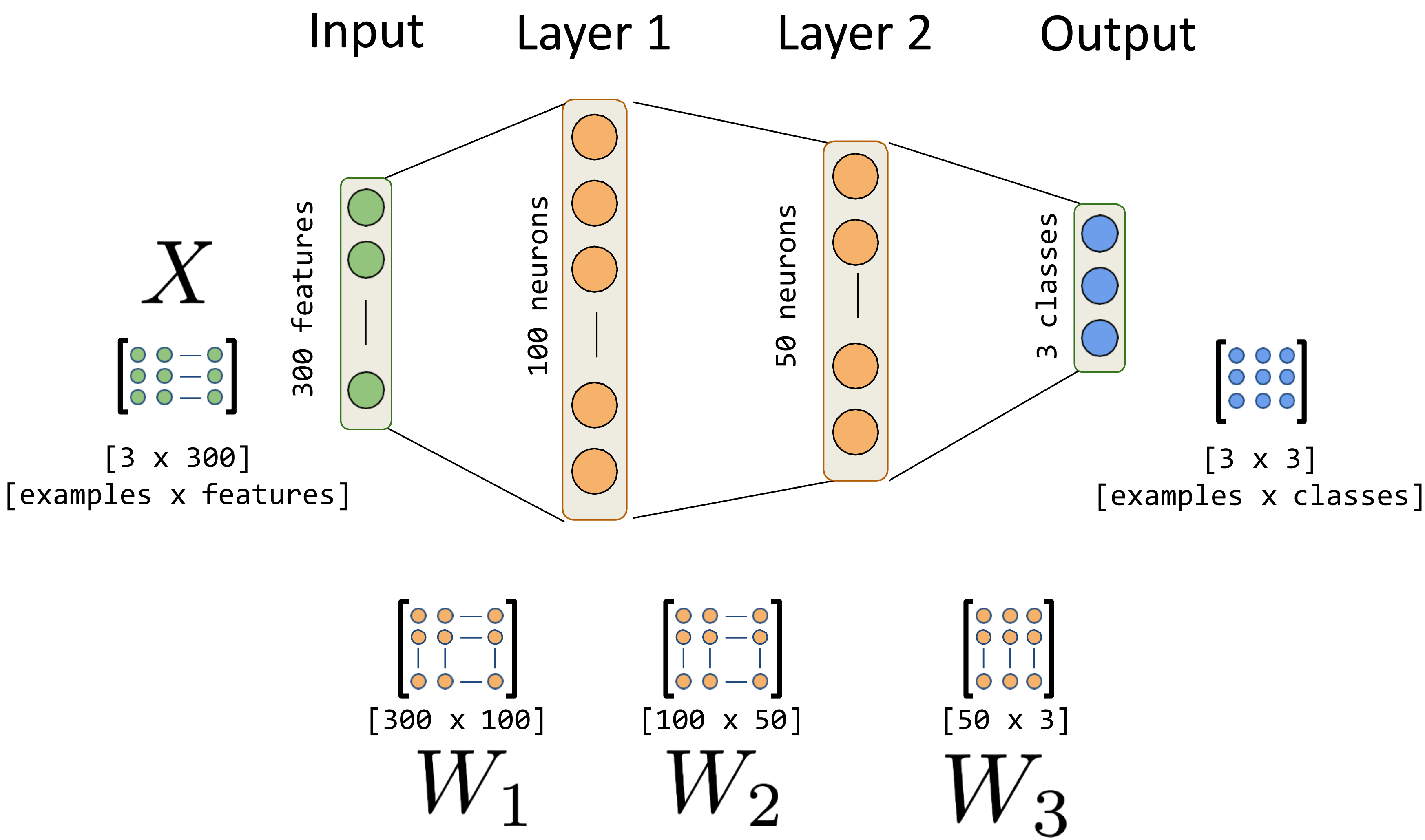
Neural Network



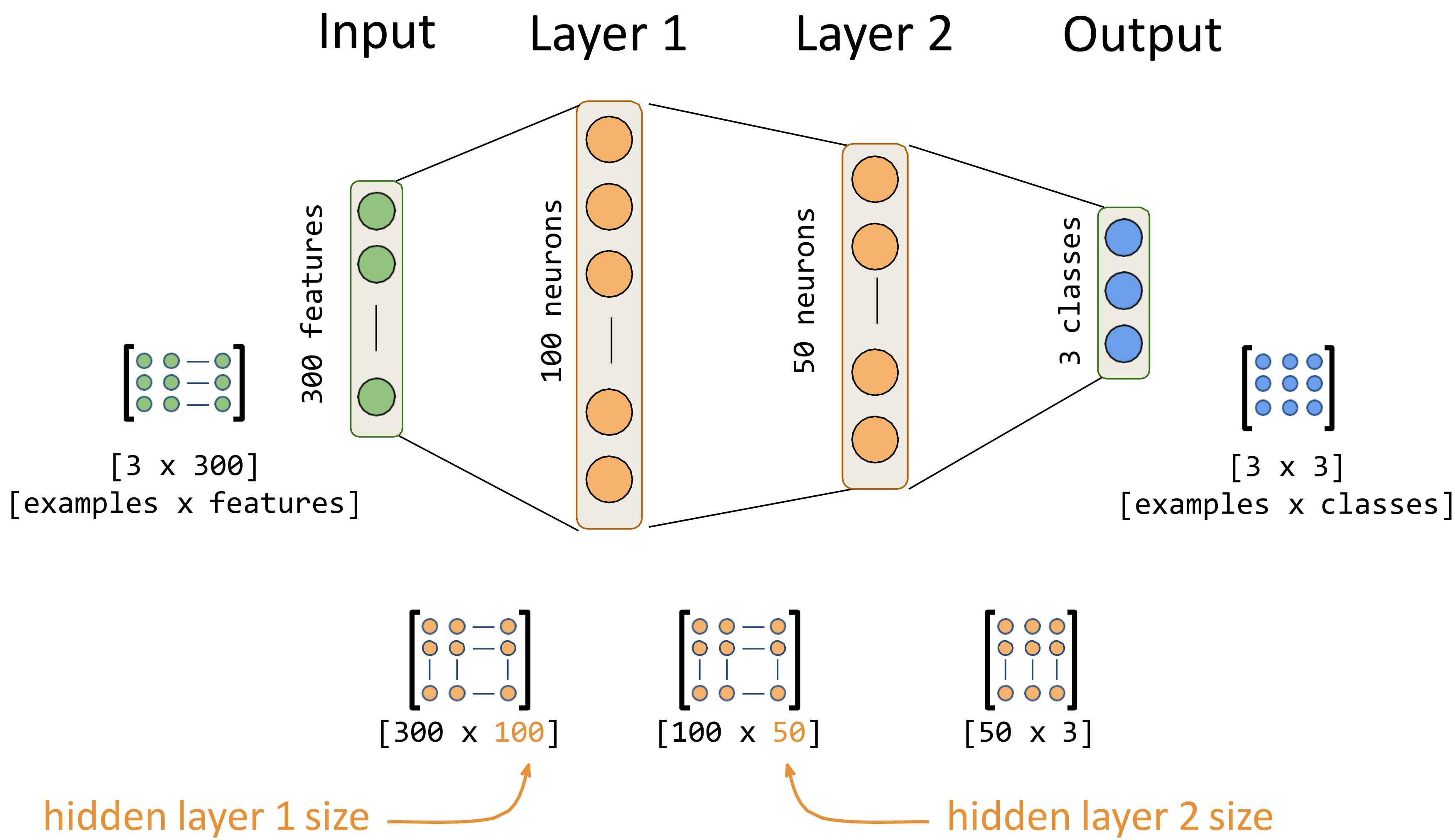
Neural Network



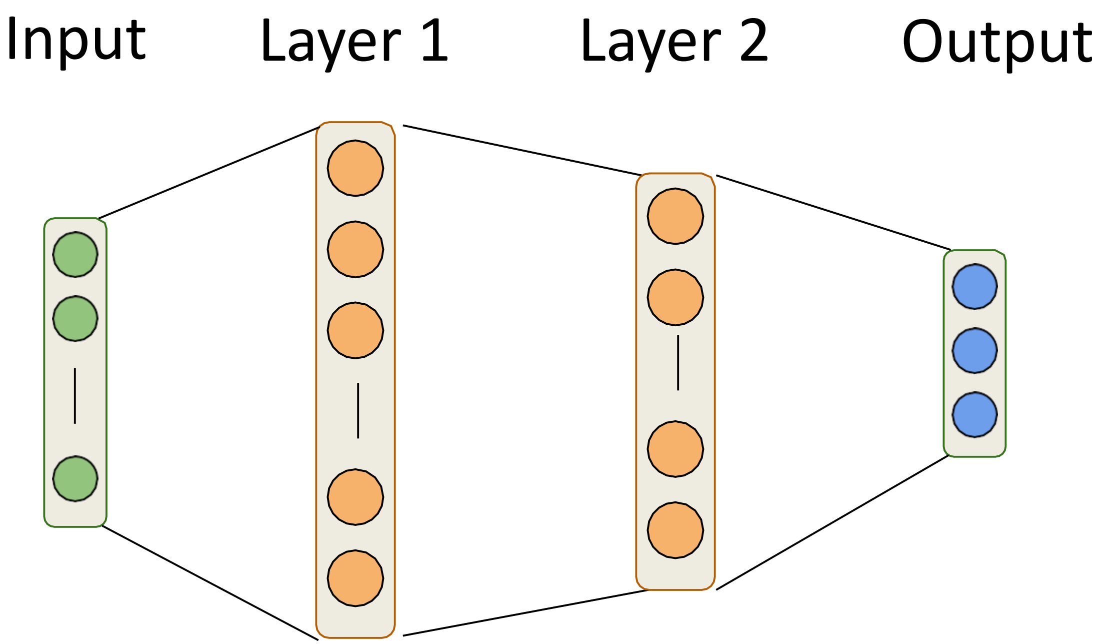
Neural Network



Neural Network



Neural Network: Optimization



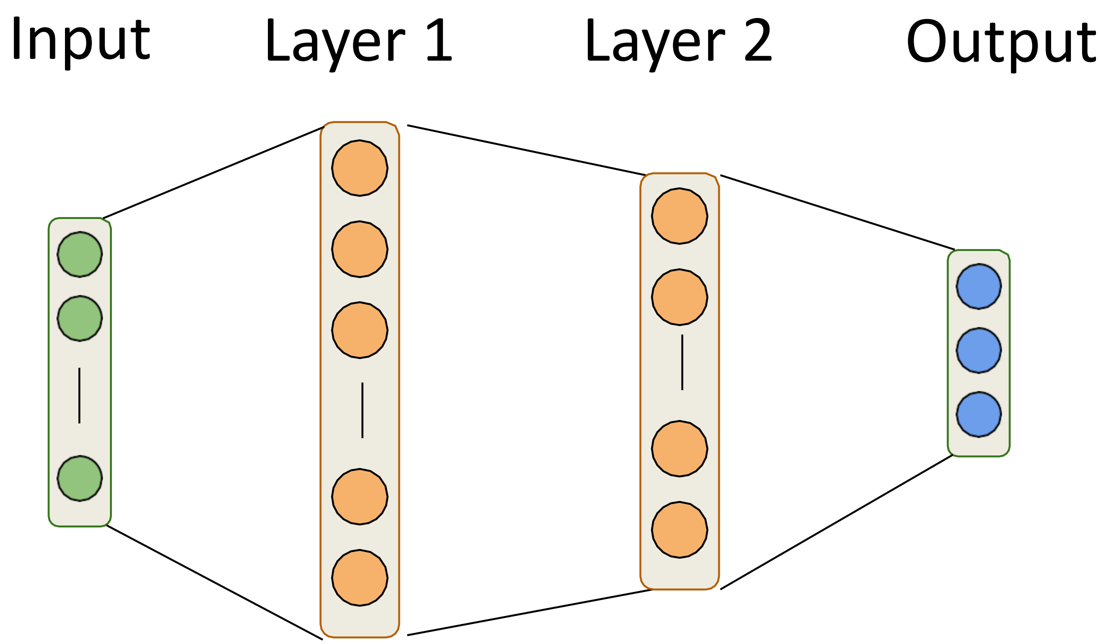
$$h_1 = \sigma(W_1 \cdot x + b_1)$$

$$h_2 = \sigma(W_2 \cdot h_1 + b_2)$$

$$f = W_3 . h_2 + b_3$$

Objective function

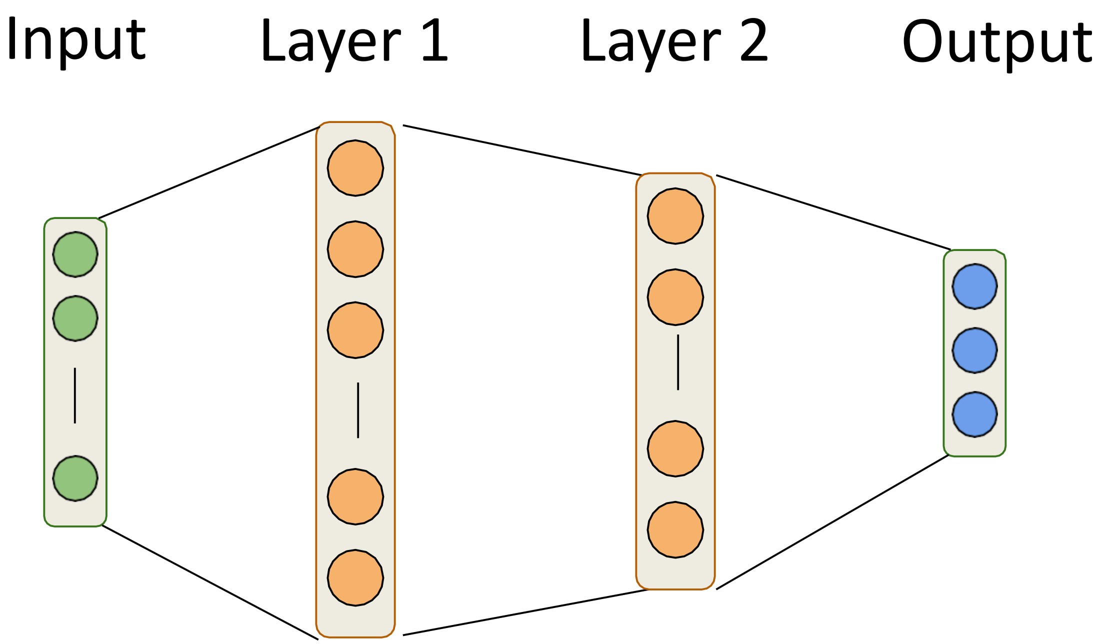
Neural Network: Optimization



$$L = -\log(f_c)$$

Cross Entropy Loss

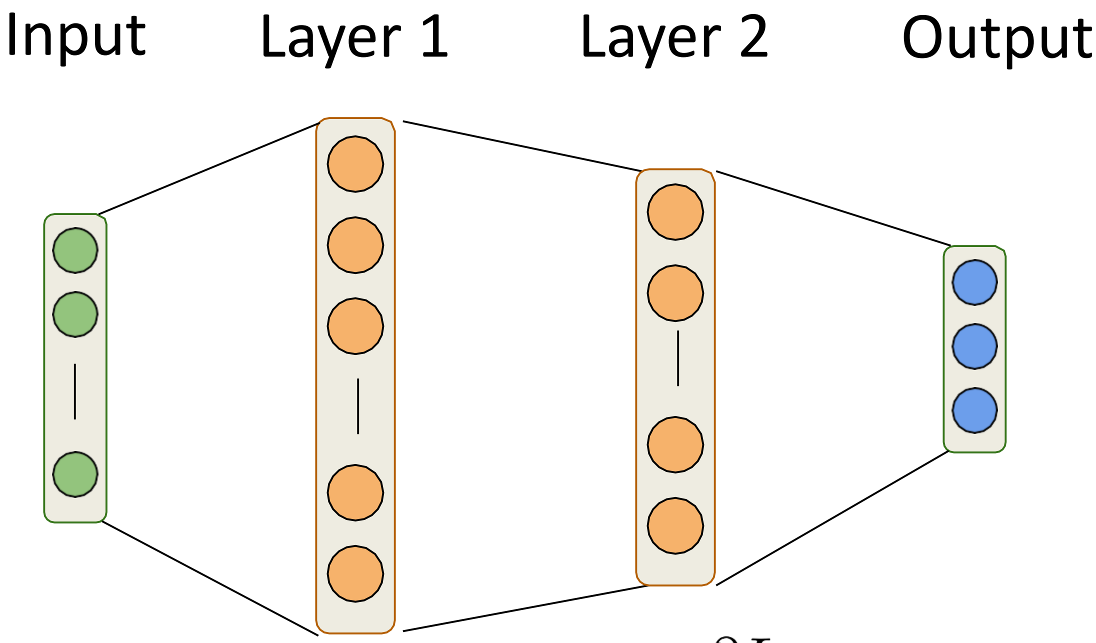
Neural Network: Optimization



Compute $\dfrac{\partial L}{\partial W_i}$ and $\dfrac{\partial L}{\partial b_i}$ using backpropagation

Optimization

Neural Network: Optimization

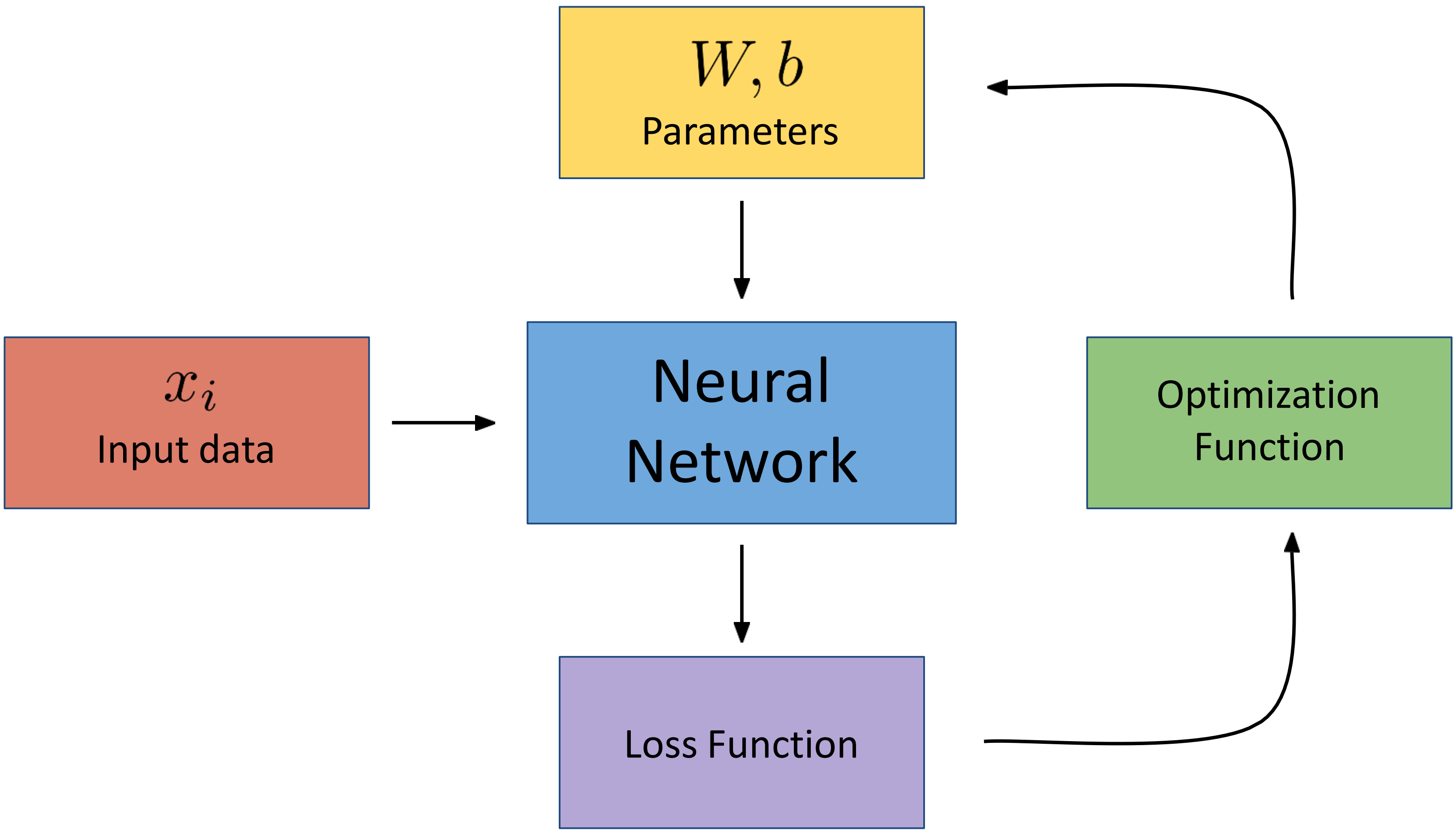


$$W_i = W_i - \eta \cdot \frac{\partial L}{\partial W_i}$$

$$b_i = b_i - \eta \cdot \frac{\partial L}{\partial b_i}$$

Parameter Update

Overall Picture



Neural Network

Let’s implement a simple two layer neural network model!

Neural Network

Recall the model definition for binary classification:

```
model = Sequential()
model.add(Dense(2, input_shape=(2,)))
model.add(Softmax())

sgd_optimizer = optimizers.SGD(lr=0.01)
model.compile(loss="categorical_crossentropy",
              optimizer=sgd_optimizer,
              metrics=['acc'])
model.summary()
```

Neural Network

Recall the model definition for binary classification:

```
model = Sequential()
model.add(Dense(2, input_shape=(2,)))
model.add(Softmax())

sgd_optimizer = optimizers.SGD(lr=0.01)
model.compile(loss="categorical_crossentropy",
              optimizer=sgd_optimizer,
              metrics=['acc'])
model.summary()
```

Just need to add a hidden layer here! (mostly)

Neural Network

Model definition

```
model = Sequential()
model.add(Dense(100, input_shape=(2,), activation='relu'))
model.add(Dense(3))
model.add(Softmax())

model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
model.summary()
```

Neural Network

Model definition

```
model = Sequential()
model.add(Dense(100, input_shape=(2,), activation='relu'))
model.add(Dense(3))
model.add(Softmax())

model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
model.summary()
```

Hidden layer

Neural Network

Model definition

```
model = Sequential()
model.add(Dense(100, input_shape=(2,), activation='relu'))
model.add(Dense(3))
model.add(Softmax())

model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
model.summary()
```

100 neurons in the hidden layer

Neural Network

Model definition

```
model = Sequential()
model.add(Dense(100, input_shape=(2,), activation='relu'))
model.add(Dense(3))
model.add(Softmax())

model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
model.summary()
```

Still only two input features

Neural Network

Model definition

```
model = Sequential()
model.add(Dense(100, input_shape=(2,), activation='relu'))
model.add(Dense(3))
model.add(Softmax())

model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
model.summary()
```

Activation function for the neurons in this layer

Neural Network

Model definition

```
model = Sequential()
model.add(Dense(100, input_shape=(2,), activation='relu'))
model.add(Dense(3))
model.add(Softmax())

model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
model.summary()
```

Activation function for the neurons in this layer

All neurons in a single layer conventionally have the same activation

Neural Network

Model definition

```
model = Sequential()
model.add(Dense(100, input_shape=(2,), activation='relu'))
model.add(Dense(3))  Output layer
model.add(Softmax())

model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
model.summary()
```

Neural Network

Model definition

```
model = Sequential()
model.add(Dense(100, input_shape=(2,), activation='relu'))
model.add(Dense(3))
model.add(Softmax())

model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
model.summary()
```

We are using the *Adam* optimizer here instead of *SGD*, since it works much better is the majority of the cases.

Neural Network

Model Learning Curve

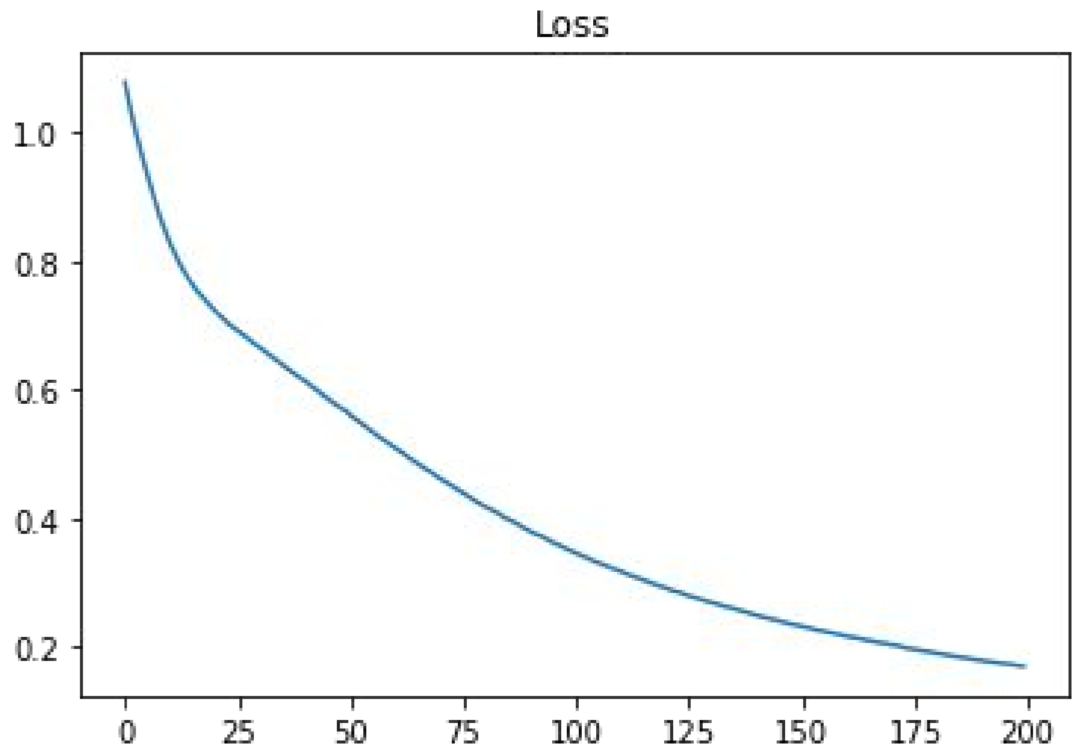
```
history = model.fit(X, y_probs, epochs=200, verbose=False)
plt.figure()
plt.plot(history.history['acc'])
plt.title("Accuracy")
plt.ylim((0.0, 1.01))

plt.figure()
plt.plot(history.history['loss'])
plt.title("Loss")
```

As we have seen before, the `fit` function returns the history of losses. We can plot these values to debug and analyze how our model is learning.

Neural Network

Model Learning Curve



As a general rule, your loss curve should go down with more epochs - we will learn more about this later

Neural Network

Neural network learns the boundaries