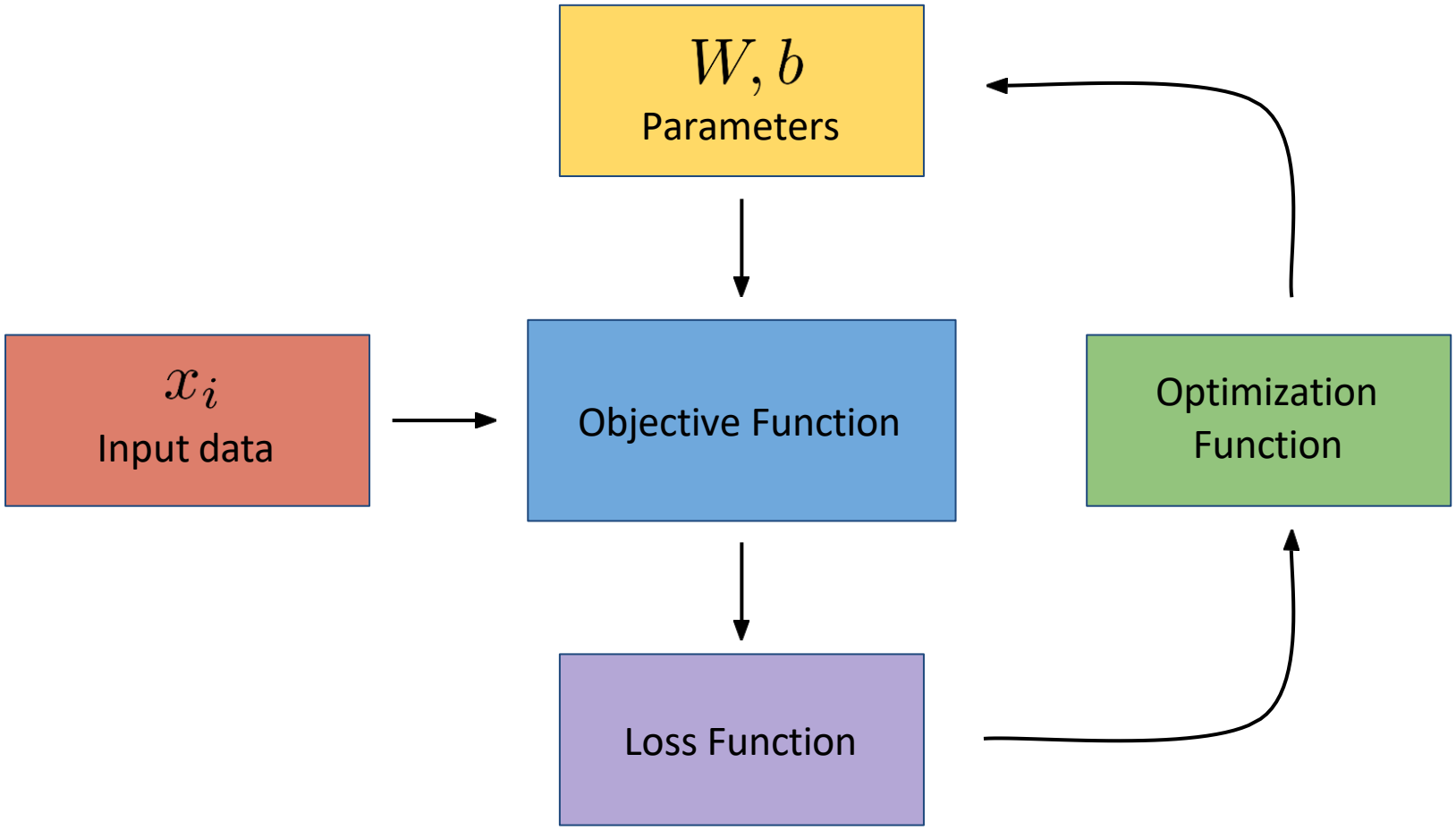
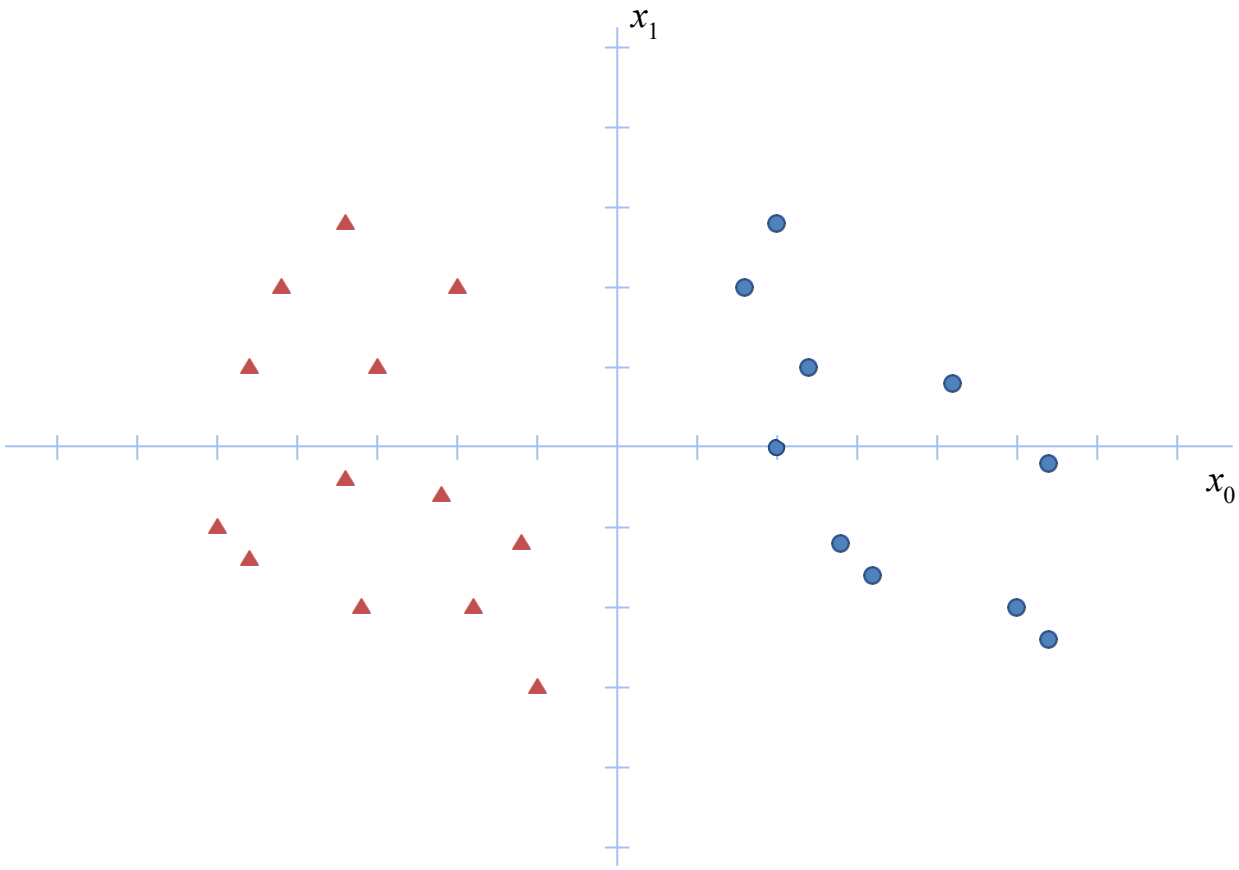
Neural Networks

Lecture # 2

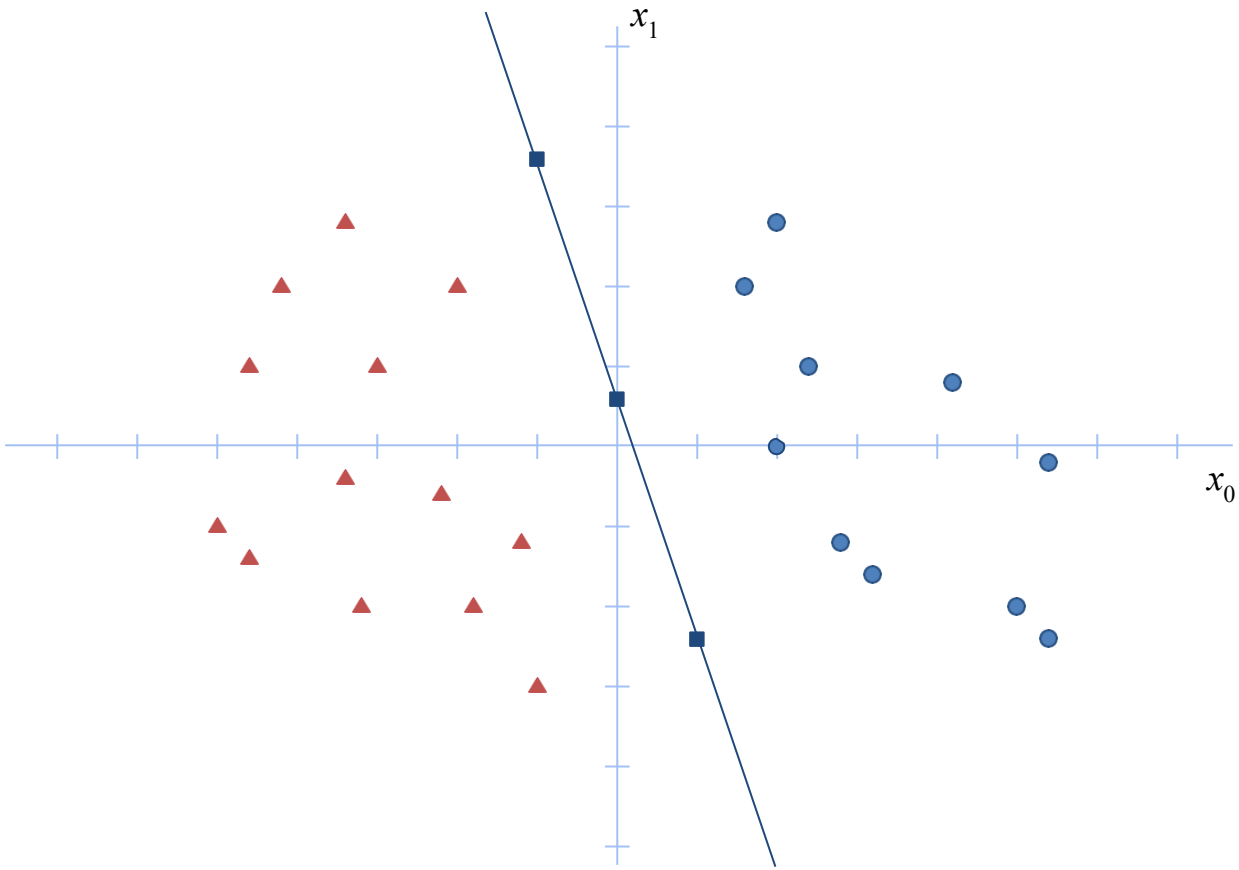
Overall Picture



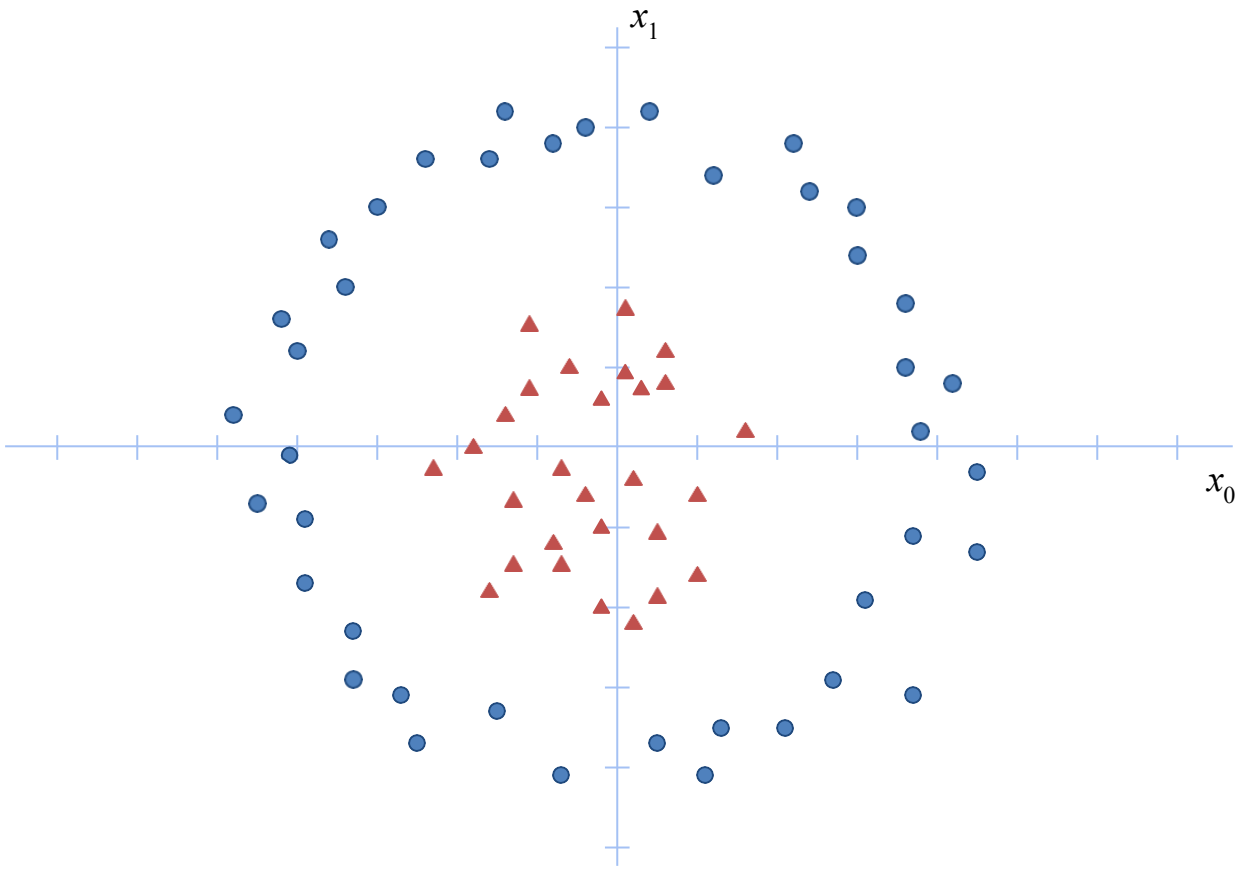
Linear Classifier Recap



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

$f(x, W, b)$

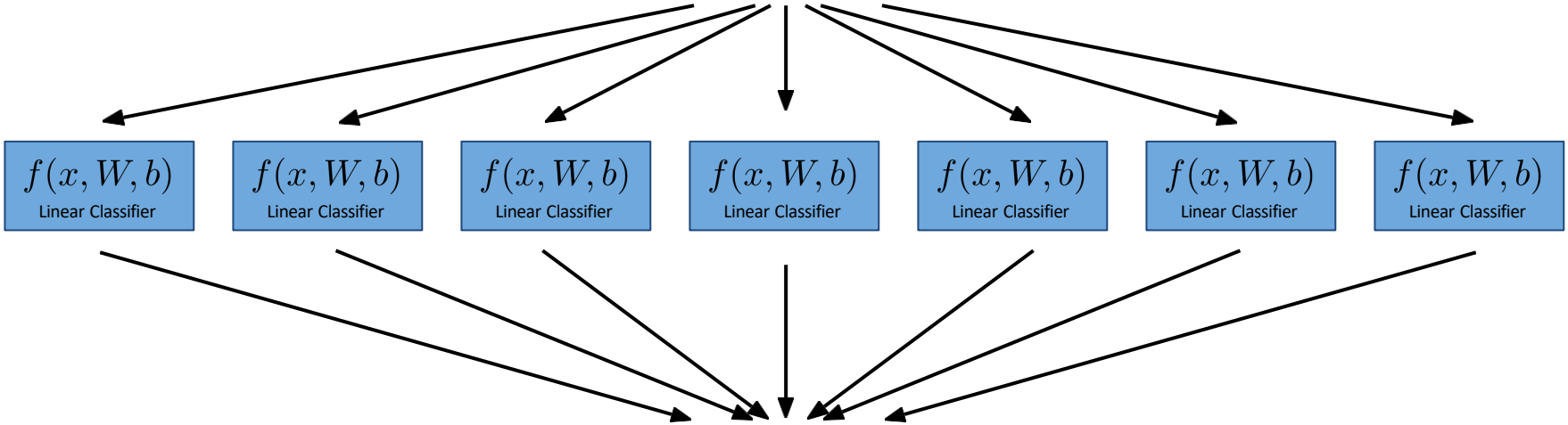
Linear Classifier

$$\begin{bmatrix} 0.3 & 1.2 \end{bmatrix}$$

Score for the car to be accident prone

Score for the car to be *NOT* accident prone

Neural Network



$$[0.3 \quad 1.2]$$

Score for the car to be accident prone

Score for the car to be *NOT* accident prone

Neural Network

Color

Speed

Acceleration

Old cars

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

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

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

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

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

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

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

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

Score for the car to be accident prone

Score for the car to be *NOT* accident prone

Neural Network

Color

Speed

Acceleration

Old cars

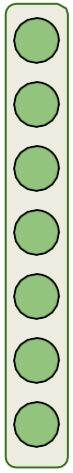
Neuron Neuron Neuron Neuron Neuron Neuron Neuron

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

Score for the car to be accident prone

Score for the car to be *NOT* accident prone

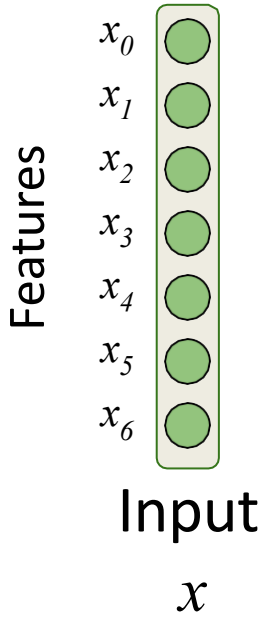
Neural Network



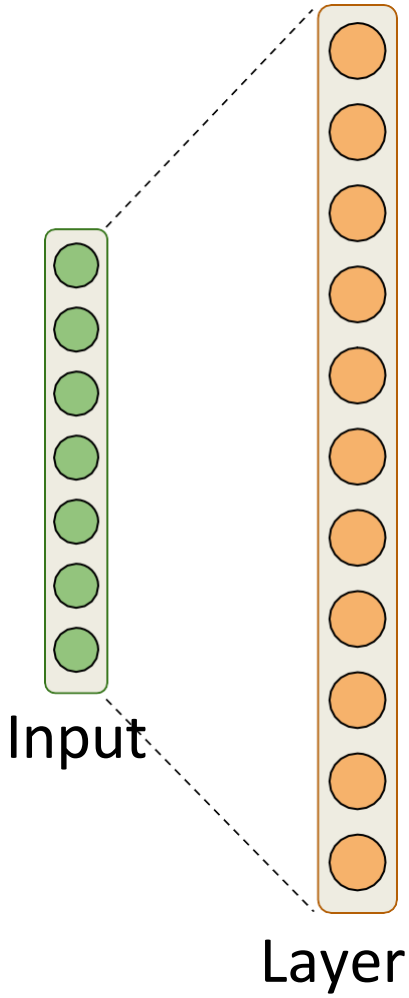
Input

x

Neural Network

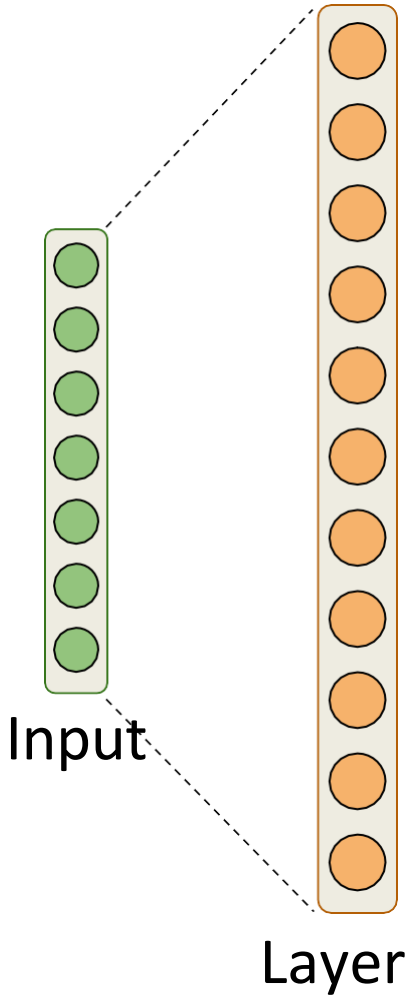


Neural Network



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

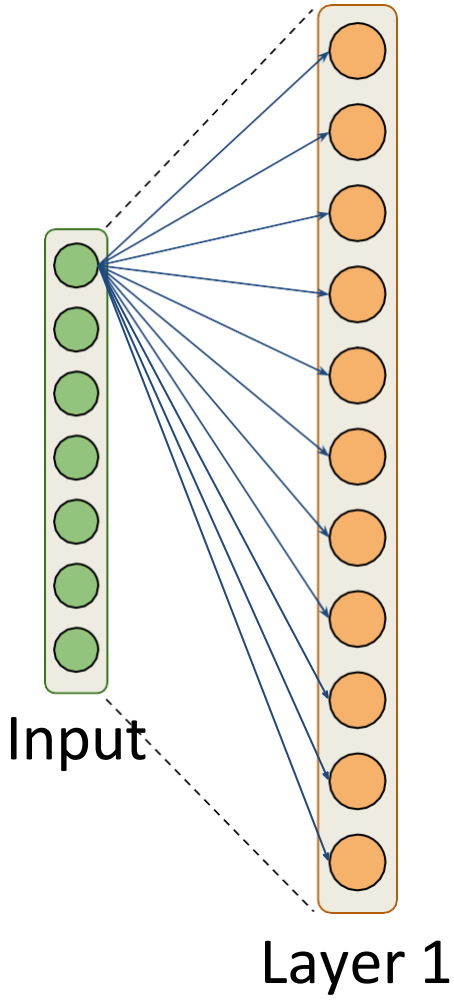
Neural Network



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

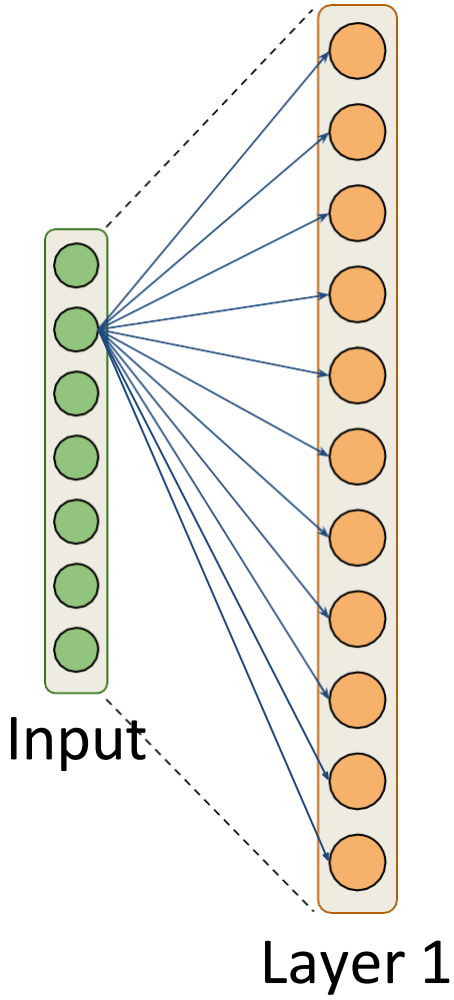
Neural Network



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

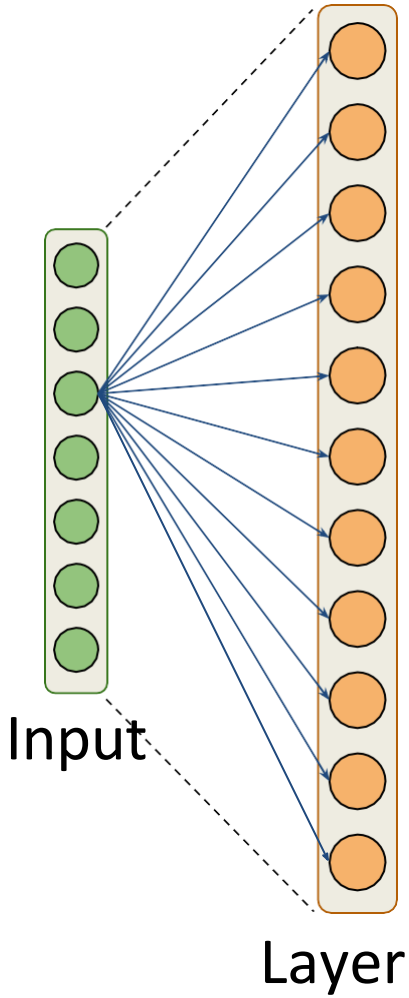
Neural Network



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

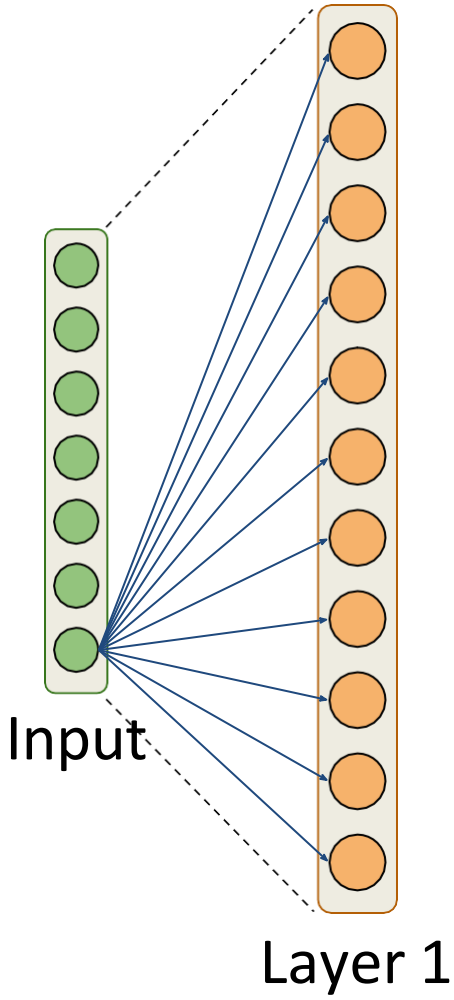
Neural Network



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

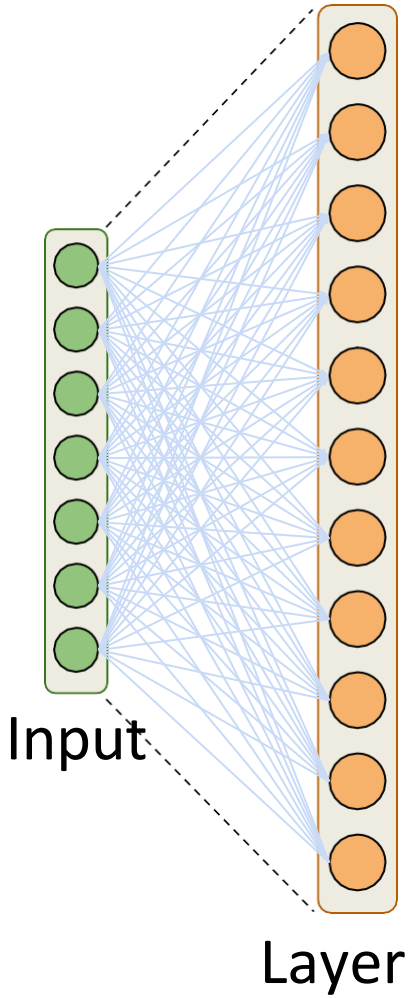
Neural Network



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

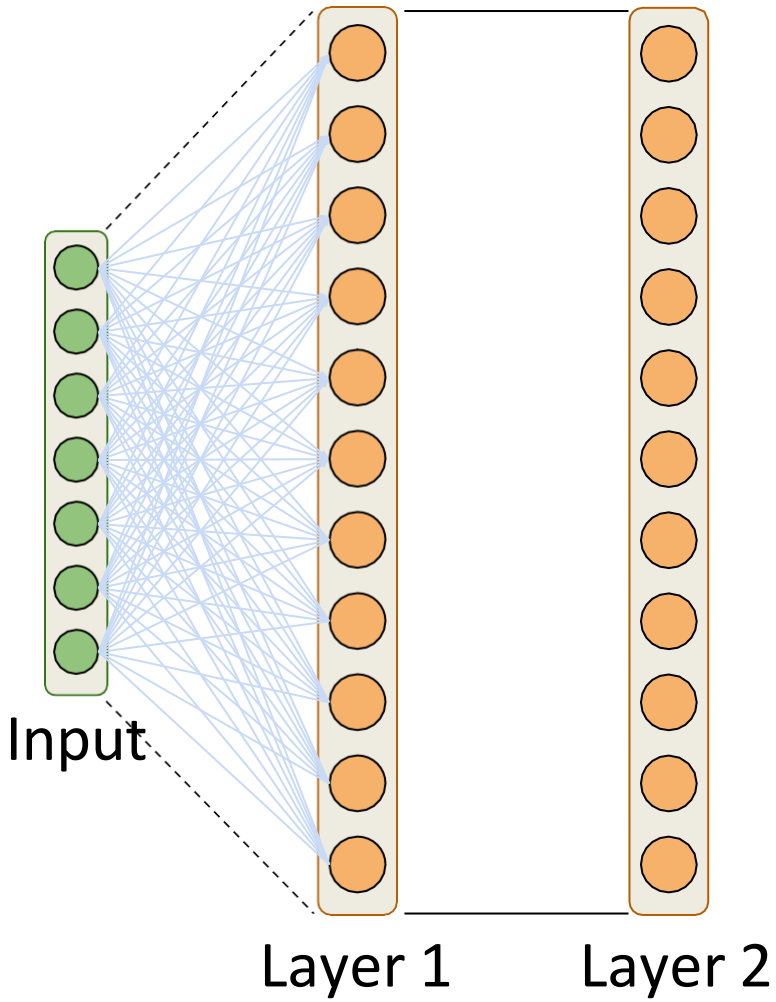
Neural Network



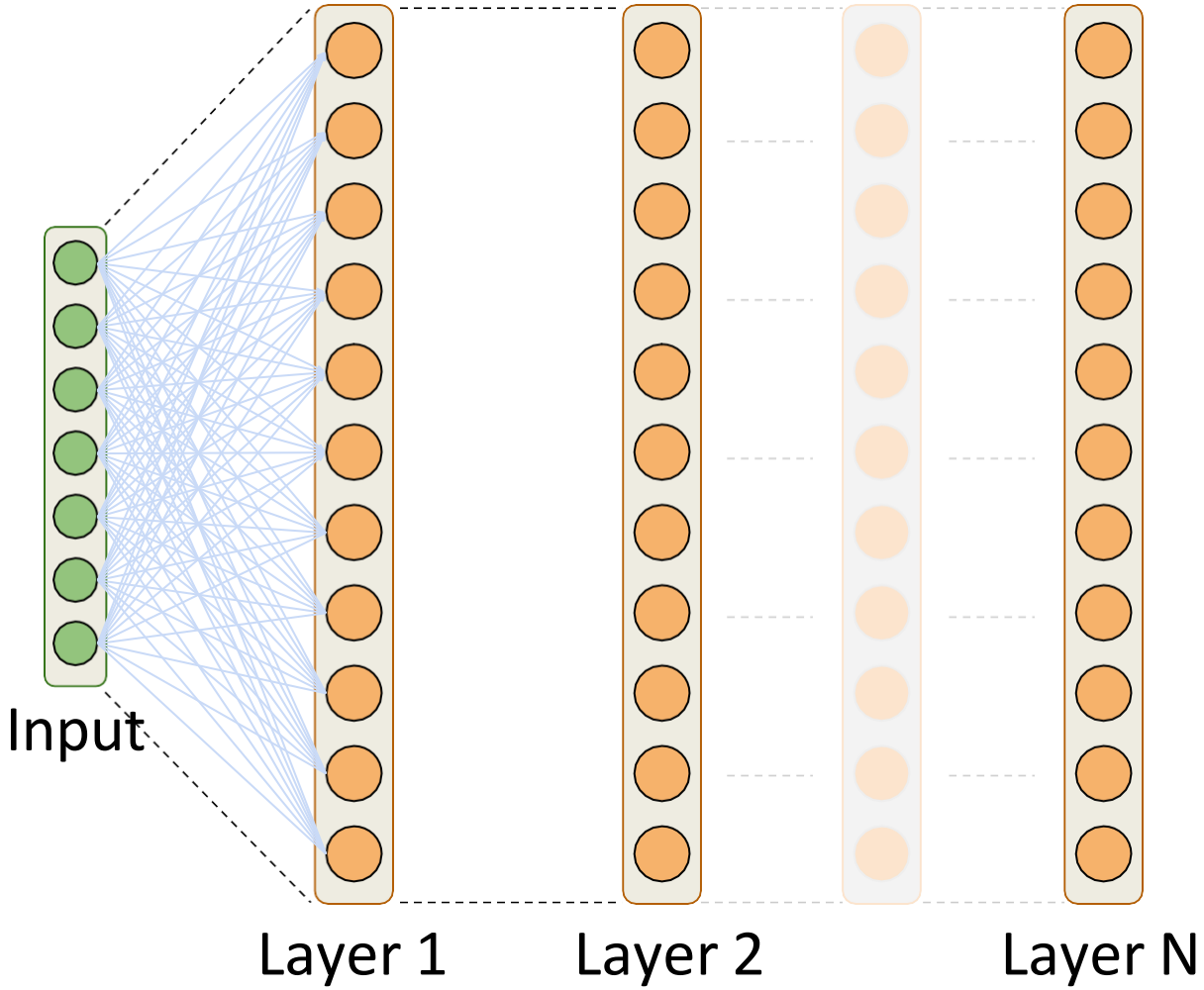
The neurons in the layer can be thought of as representing *richer features*

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

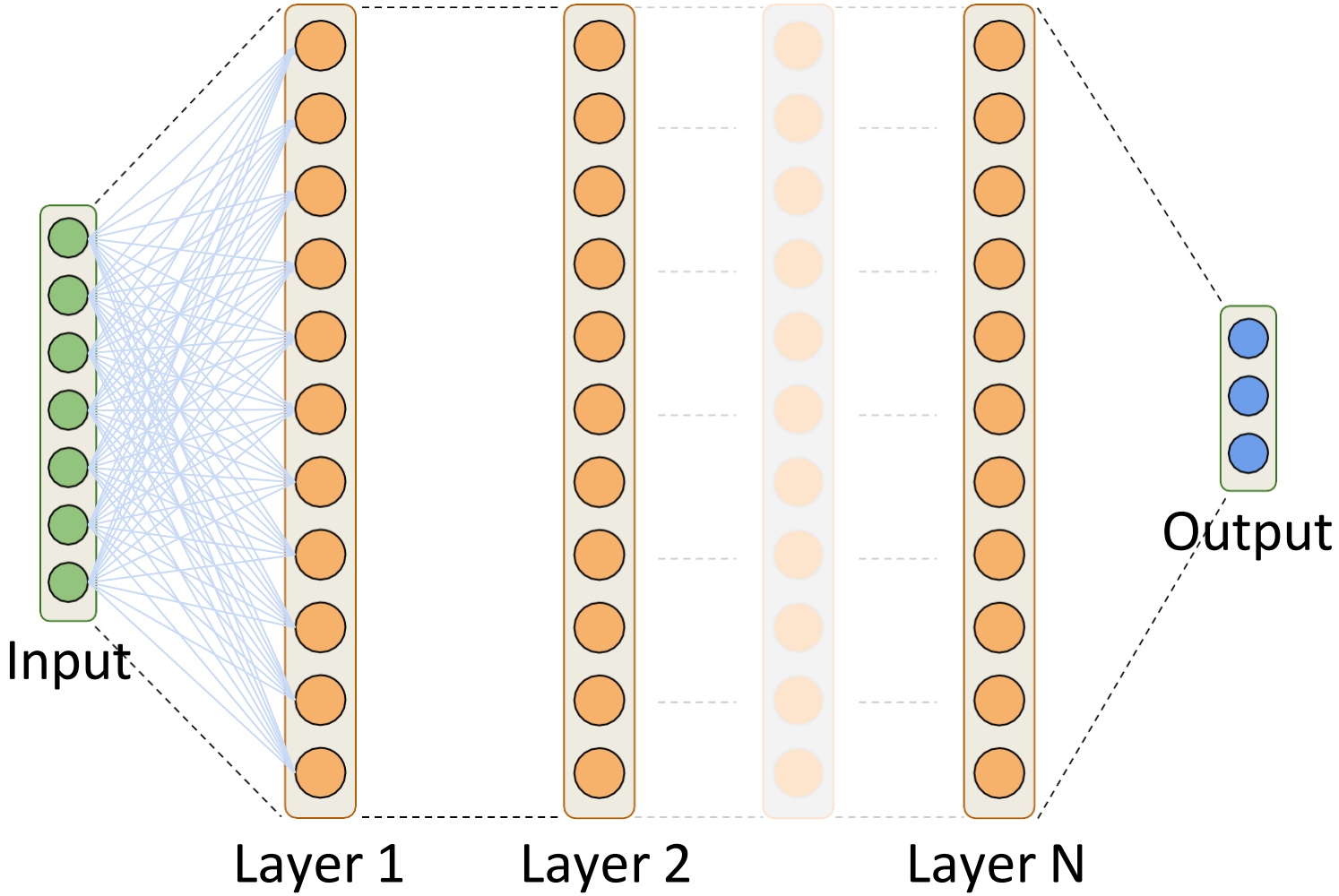
Neural Network



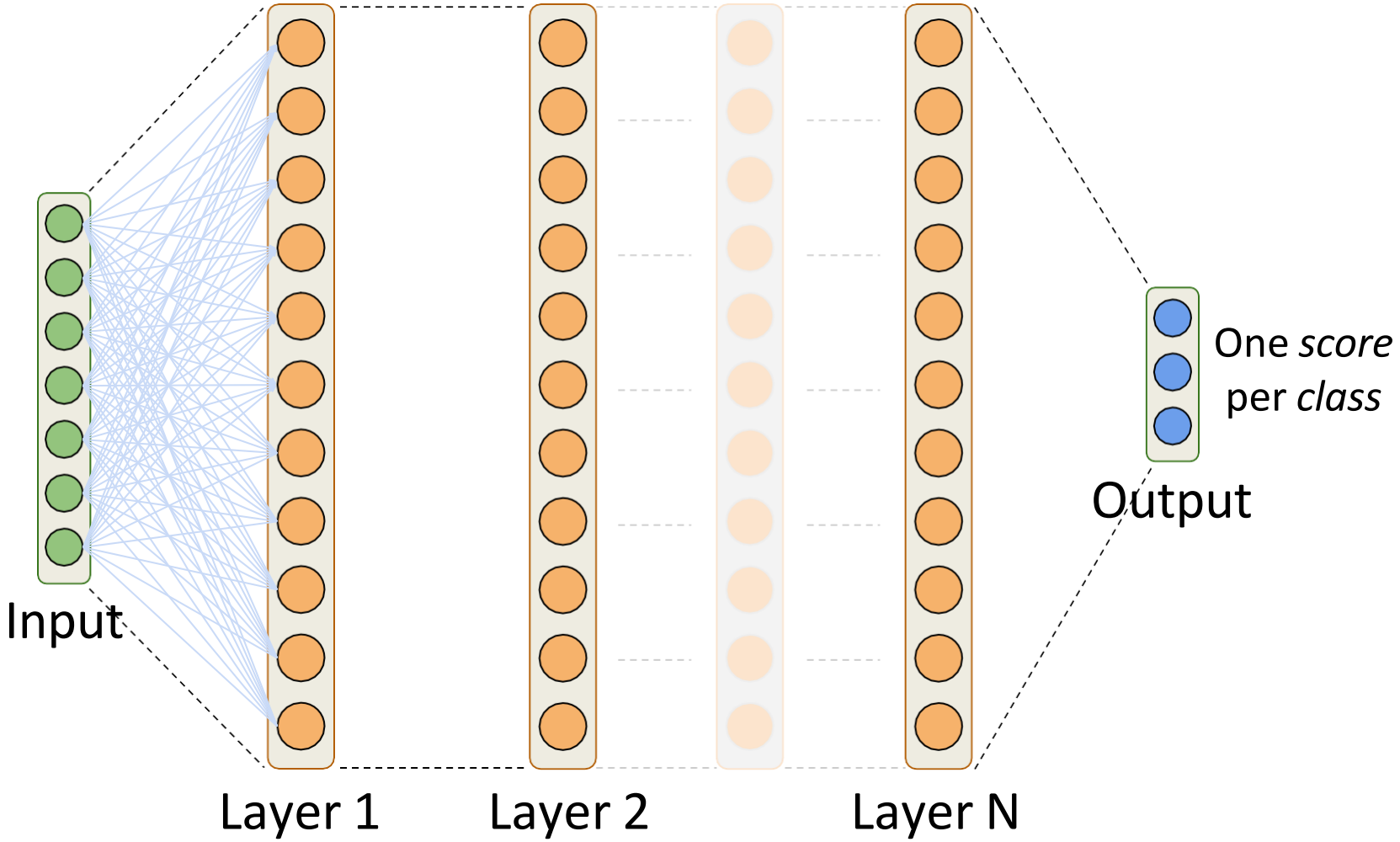
Neural Network



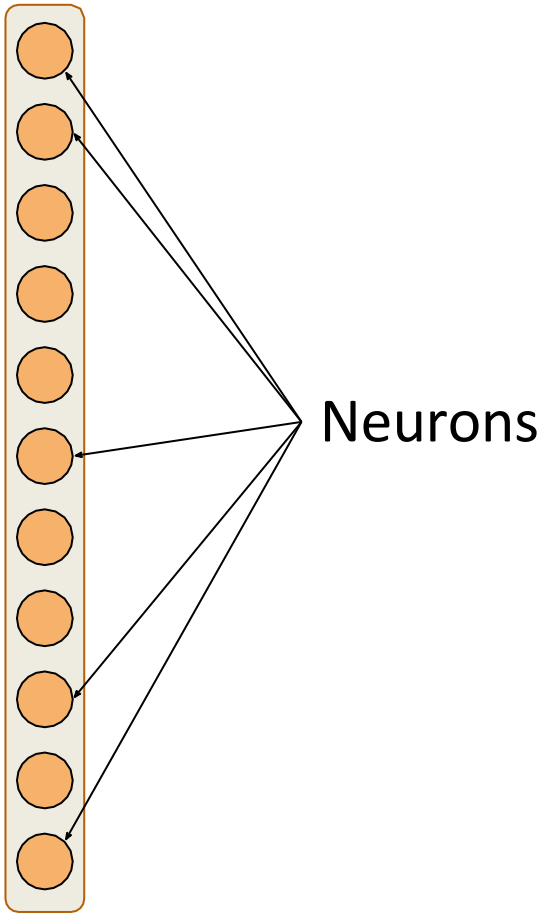
Neural Network



Neural Network

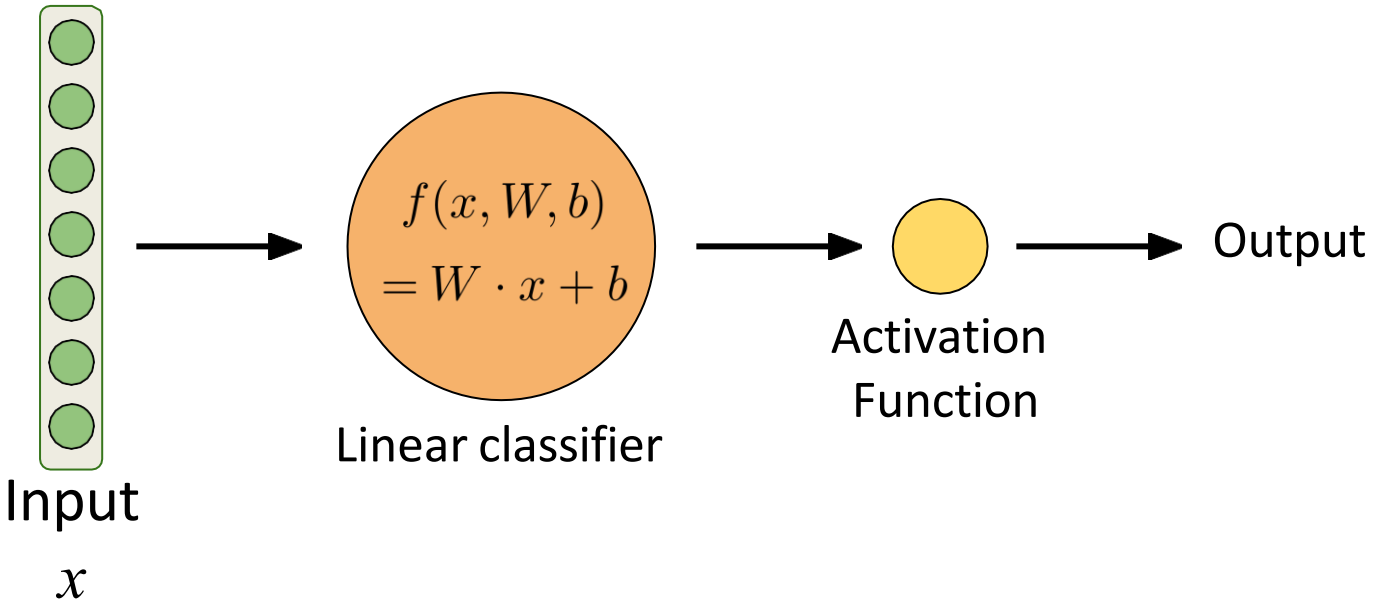


Neural Network



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

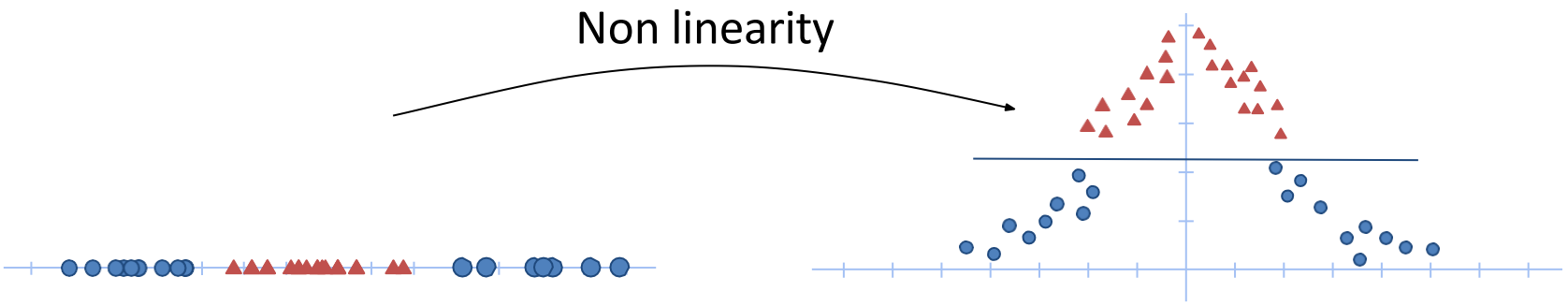
Activation Functions

- 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

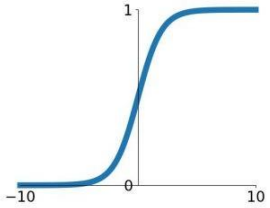
- Intuitively, a neuron looks at a particular feature of the data
- The activation after the linear classifier gives us an idea of how much the neuron “supports” the feature
- Activations also helps us map linear spaces into non-linear spaces



Activation Functions

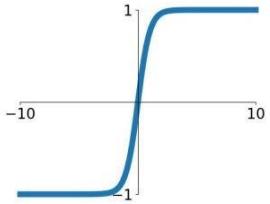
Sigmoid

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



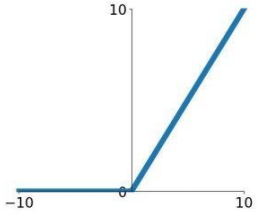
tanh

$$\tanh(x)$$



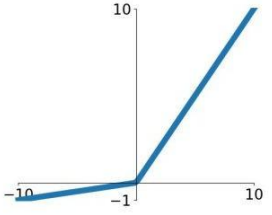
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

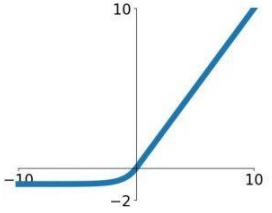


Maxout

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

ELU

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



Neural Network

- Entire network is nothing but a function:

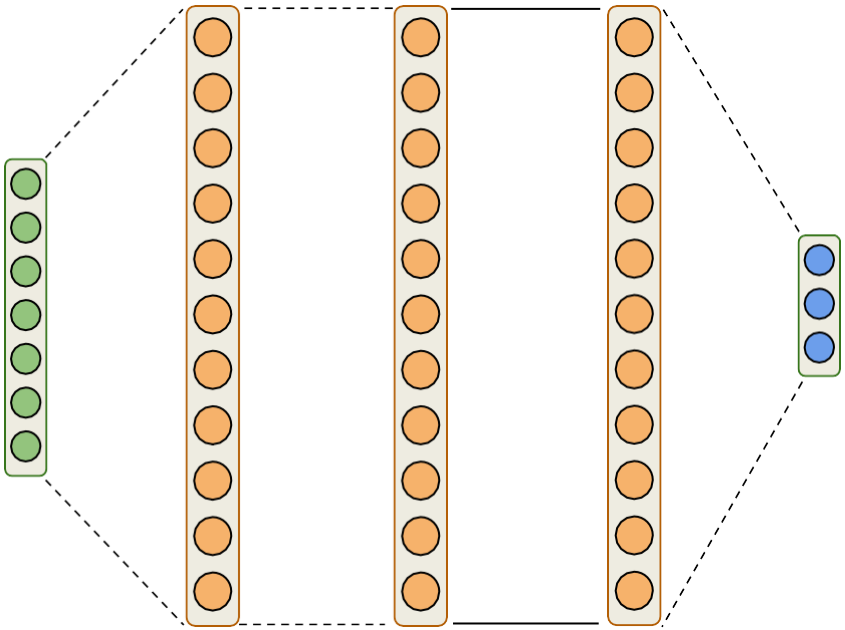
$$f = W \cdot x + b$$

Linear classifier

Neural Network

- Entire network is nothing but a function:

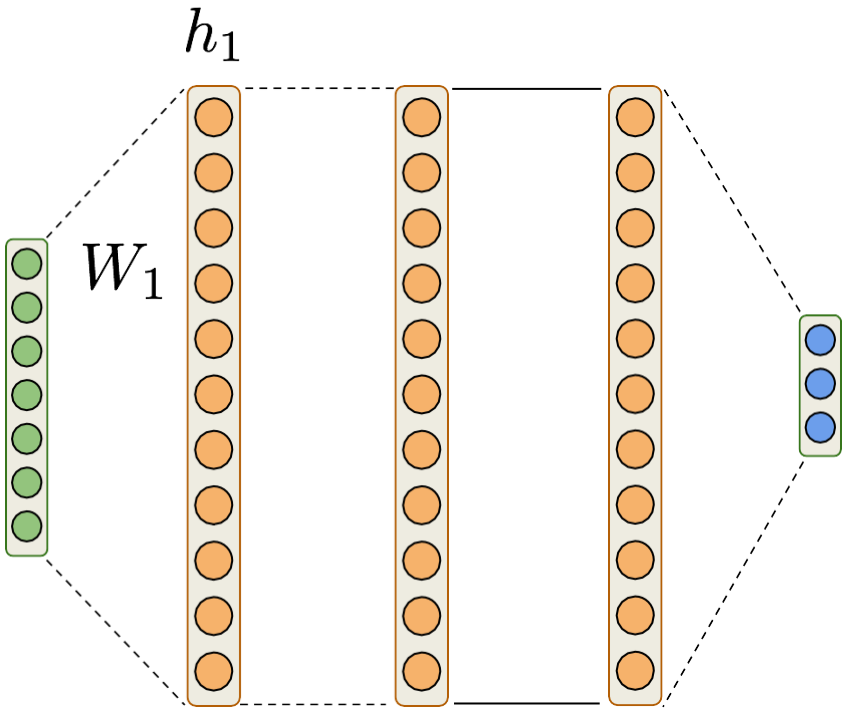
Neural network with 3 hidden layers



Neural Network

- Entire network is nothing but a function:

Neural network with 3 hidden layers

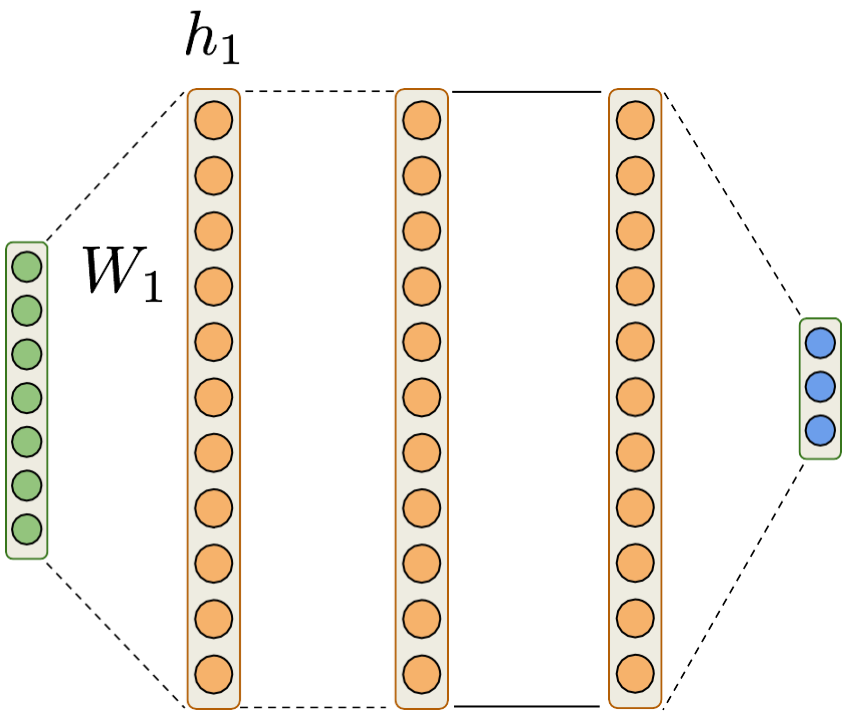


$$h_1 = \sigma(W_1 \cdot x + b_1)$$

Neural Network

- Entire network is nothing but a function:

Neural network with 3 hidden layers



Activation function

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

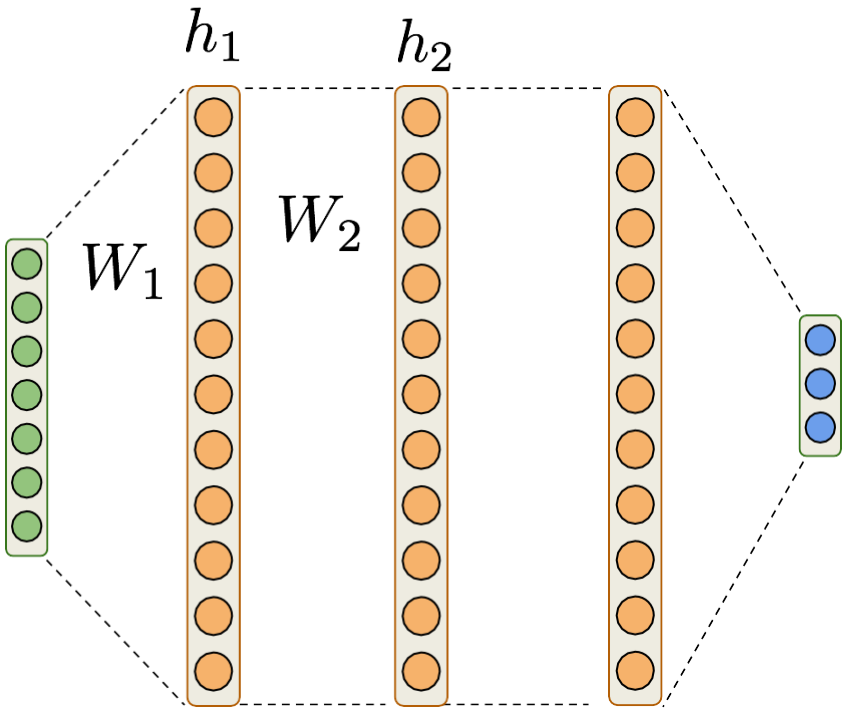
Output of linear classifier

"richer features"

Neural Network

- Entire network is nothing but a function:

Neural network with 3 hidden layers

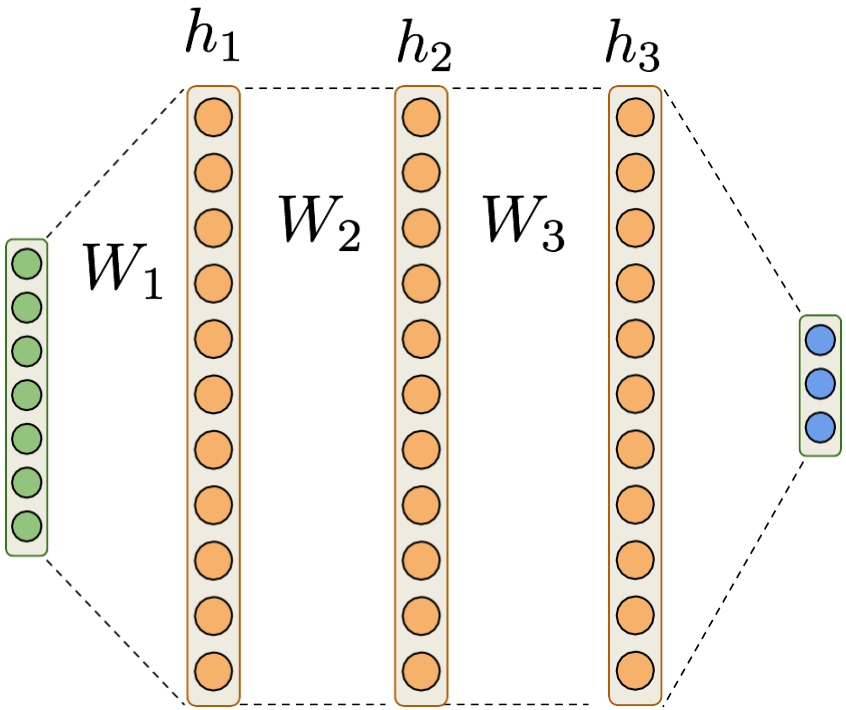


$$h_1 = \sigma(W_1 \cdot x + b_1)$$
$$h_2 = \sigma(W_2 \cdot h_1 + b_2)$$

Neural Network

- Entire network is nothing but a function:

Neural network with 3 hidden layers

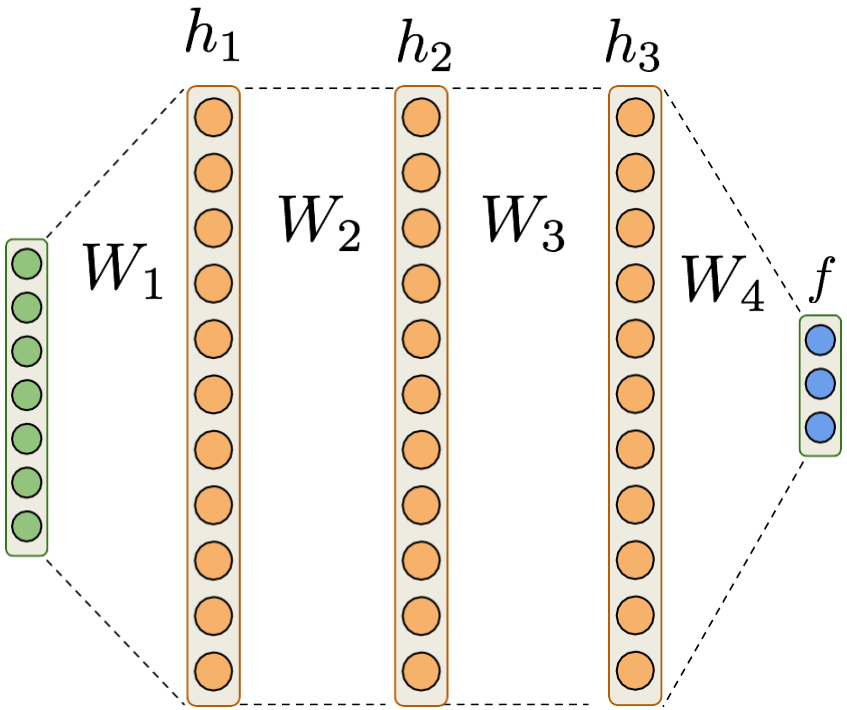


$$h_1 = \sigma(W_1 \cdot x + b_1)$$
$$h_2 = \sigma(W_2 \cdot h_1 + b_2)$$
$$h_3 = \sigma(W_3 \cdot h_2 + b_3)$$

Neural Network

- Entire network is nothing but a function:

Neural network with 3 hidden layers

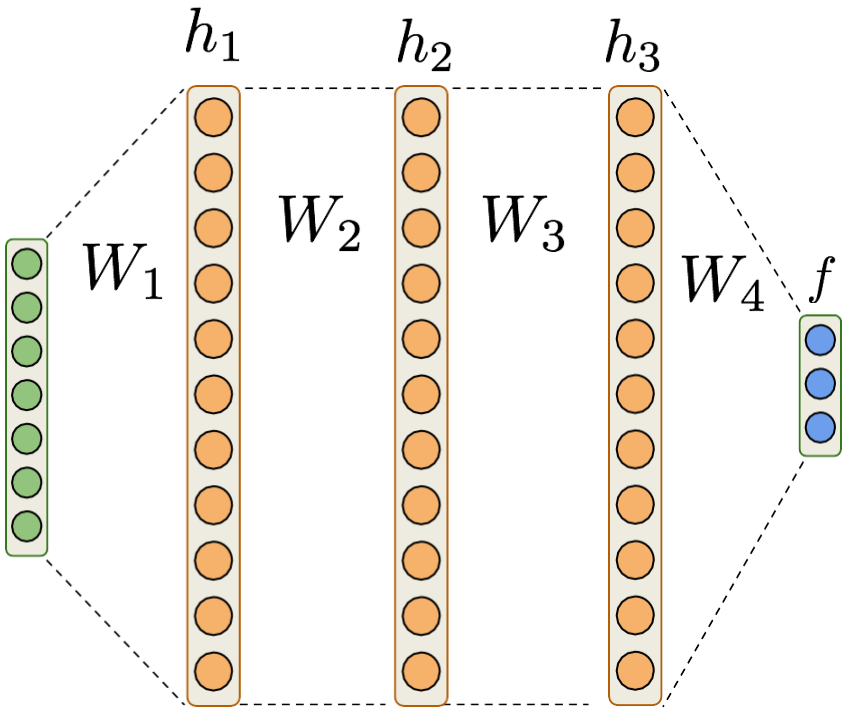


$$h_1 = \sigma(W_1 \cdot x + b_1)$$
$$h_2 = \sigma(W_2 \cdot h_1 + b_2)$$
$$h_3 = \sigma(W_3 \cdot h_2 + b_3)$$
$$f = W_4 \cdot h_3 + b_4$$

Neural Network

- Entire network is nothing but a function:

Neural network with 3 hidden layers



$$h_1 = \sigma(W_1 \cdot x + b_1)$$
$$h_2 = \sigma(W_2 \cdot h_1 + b_2)$$
$$h_3 = \sigma(W_3 \cdot h_2 + b_3)$$
$$f = W_4 \cdot h_3 + b_4$$

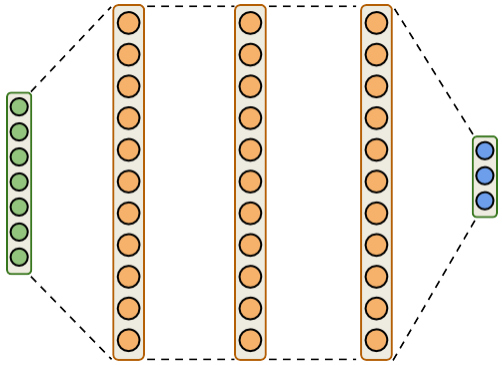
Final scores

Neural Network

- Everything else remains the same!

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

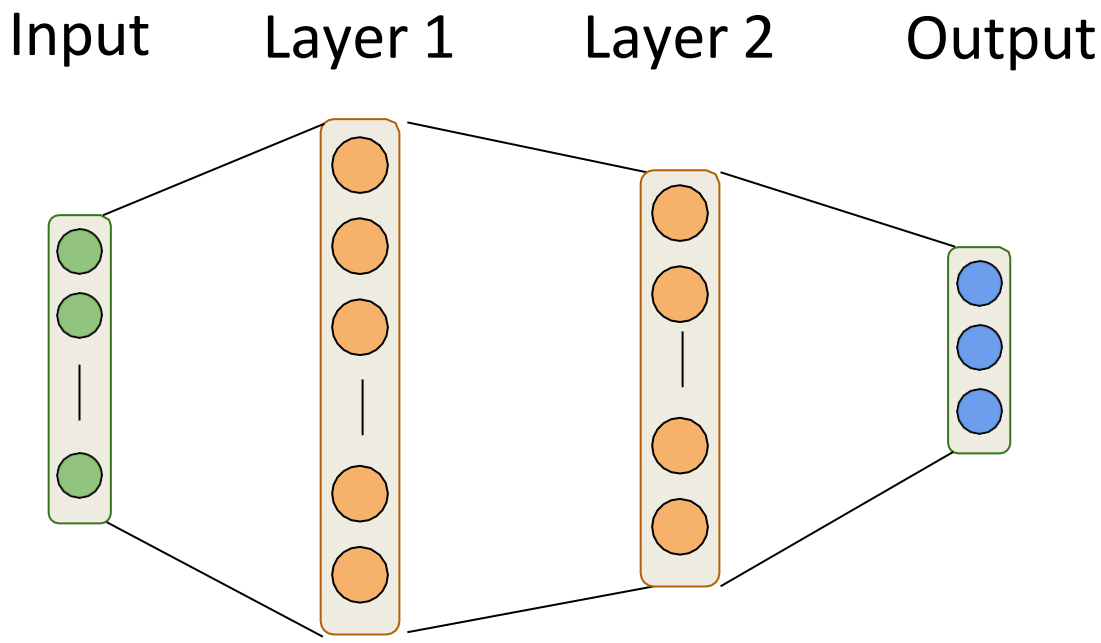
Linear classifier



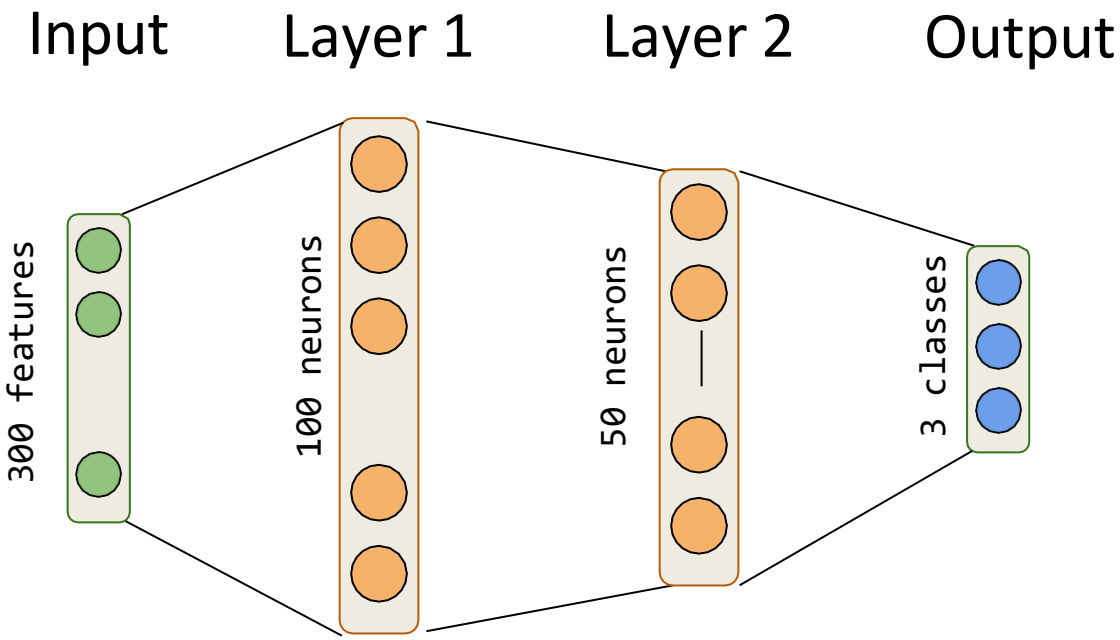
$$f = W_4 \cdot (\sigma(W_3 \cdot (\sigma(W_2 \cdot (\sigma(W_1 \cdot x + b_1)) + b_2)) + b_3)) + b_4$$

Neural network with 3 hidden layers

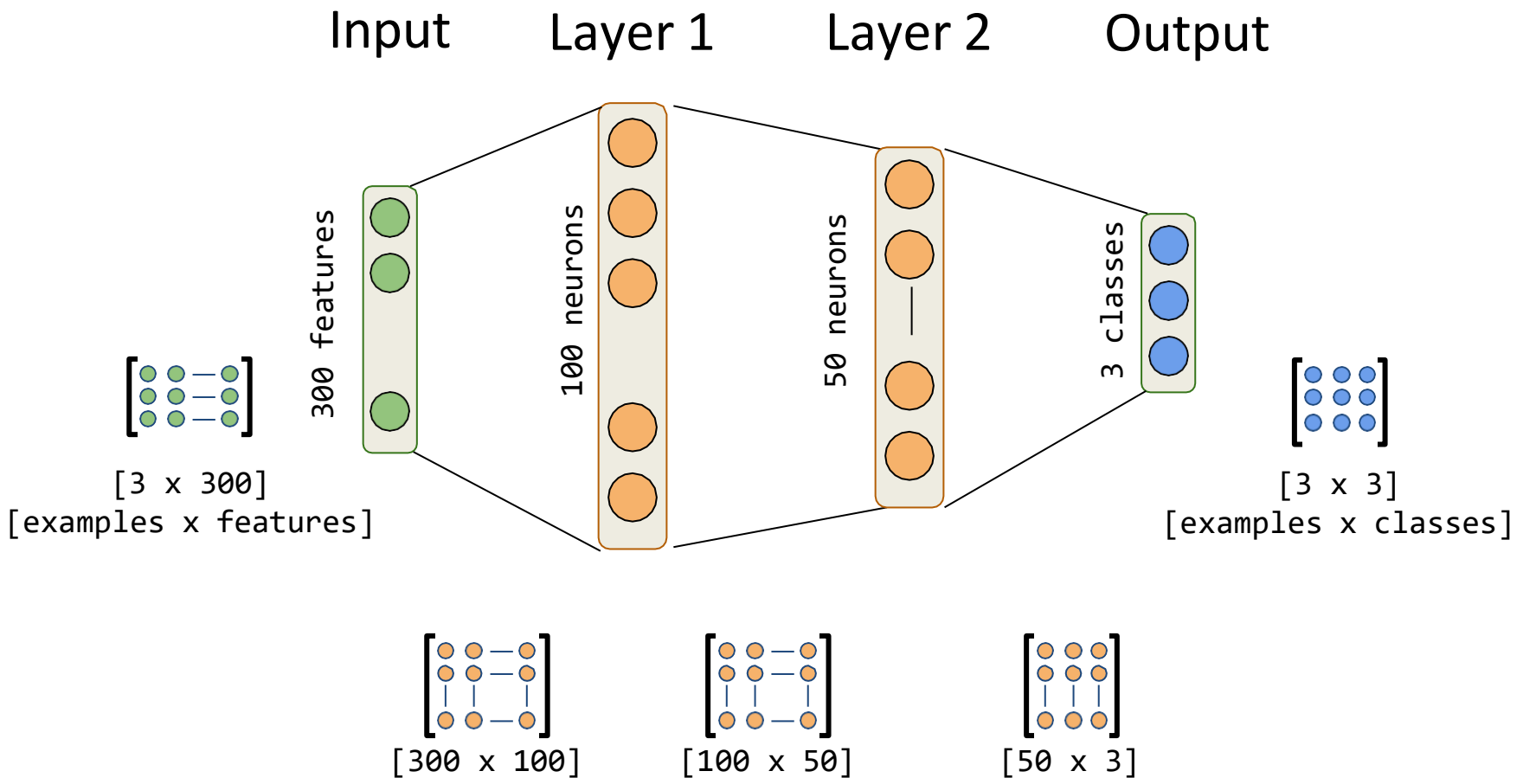
Neural Network



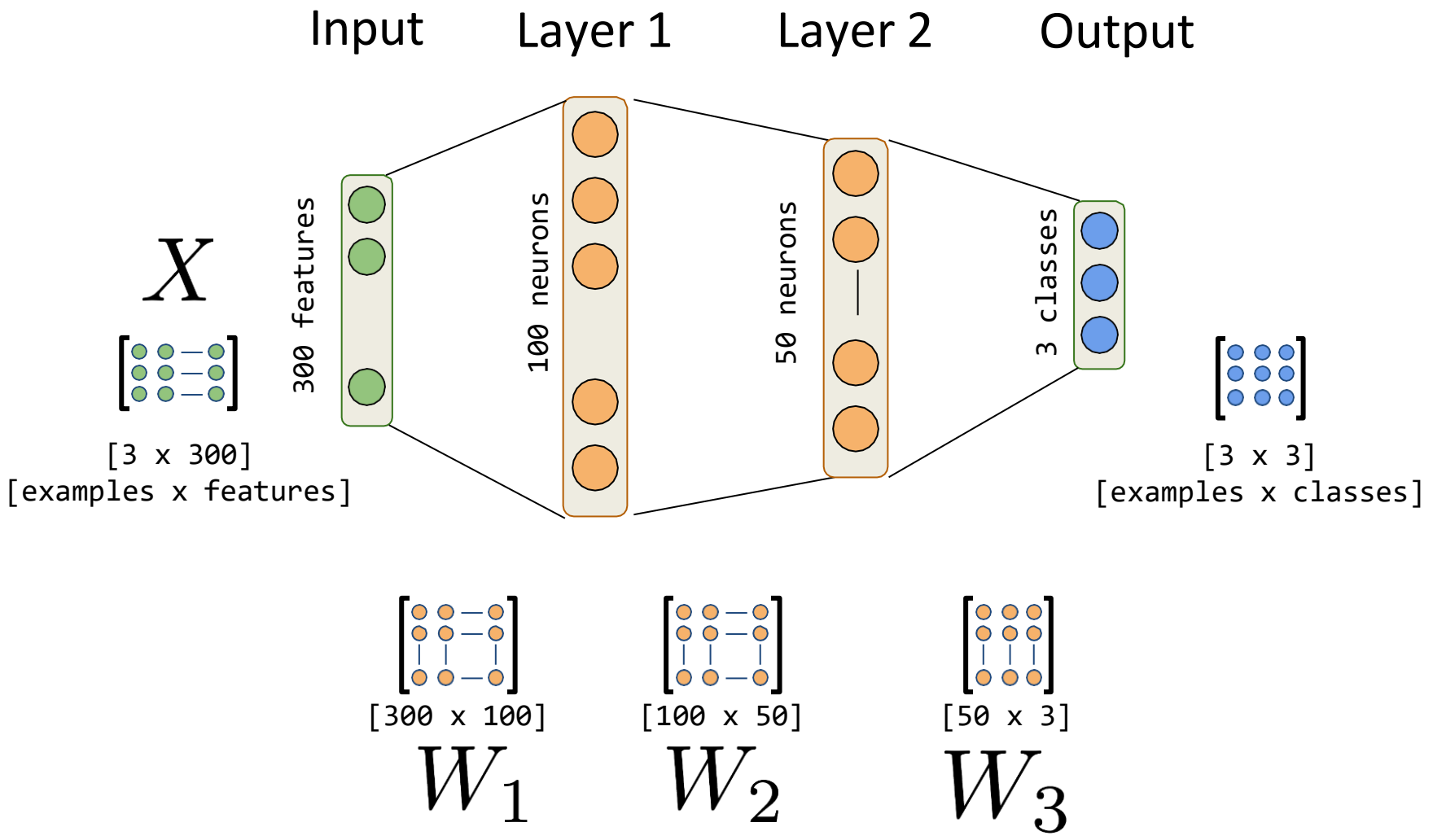
Neural Network



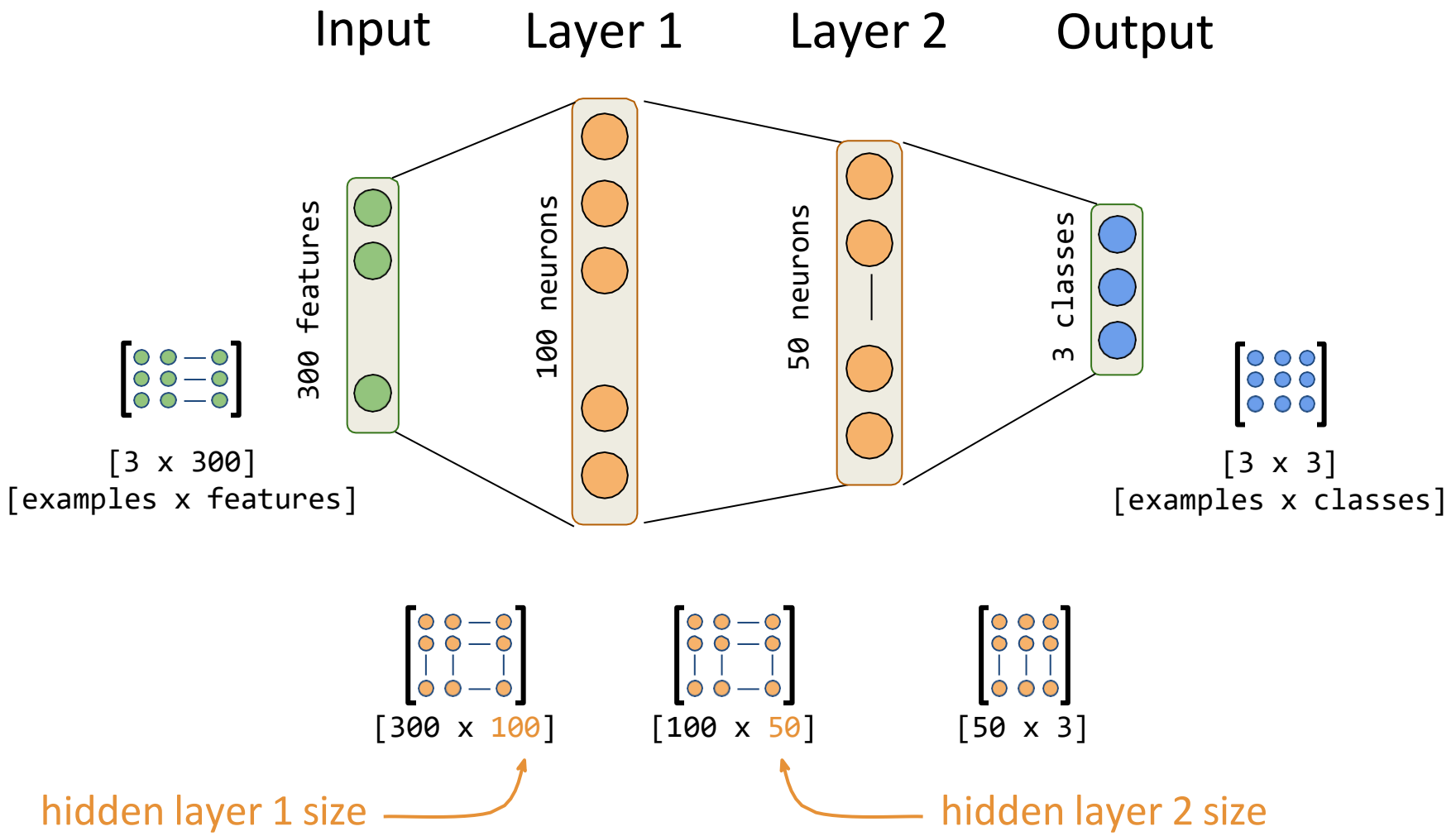
Neural Network



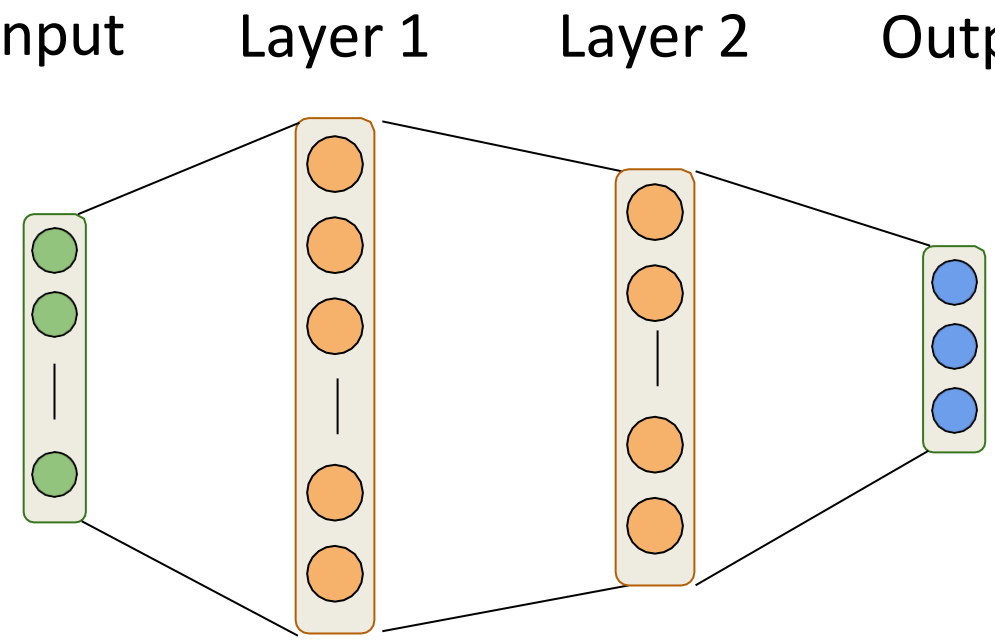
Neural Network



Neural Network



Neural Network: Optimization



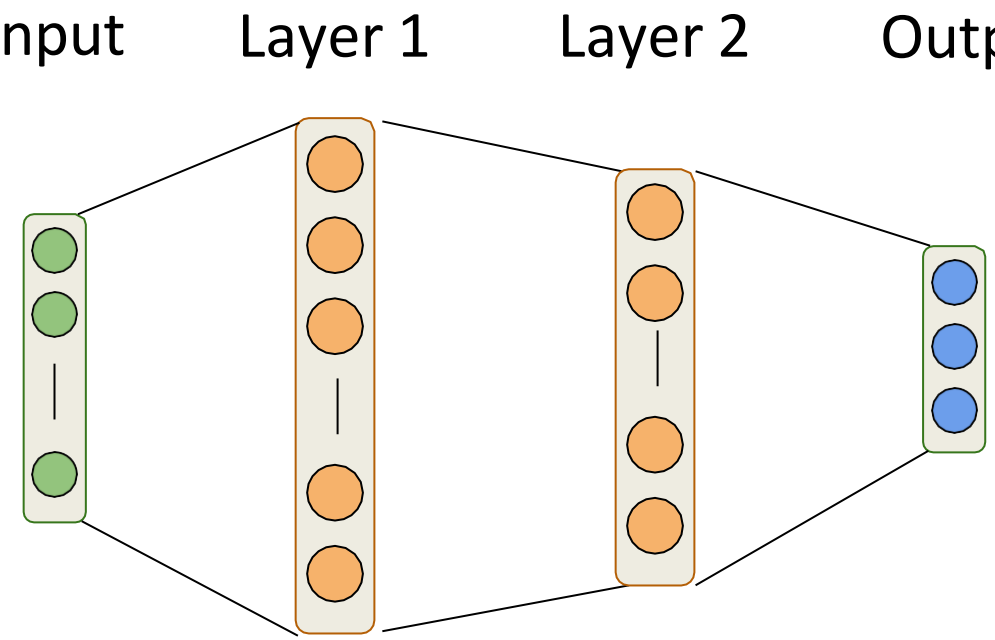
$$h_1 = \sigma(W_1 \cdot x + b_1)$$

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

$$f = W_3.h_2 + b_3$$

Objective function

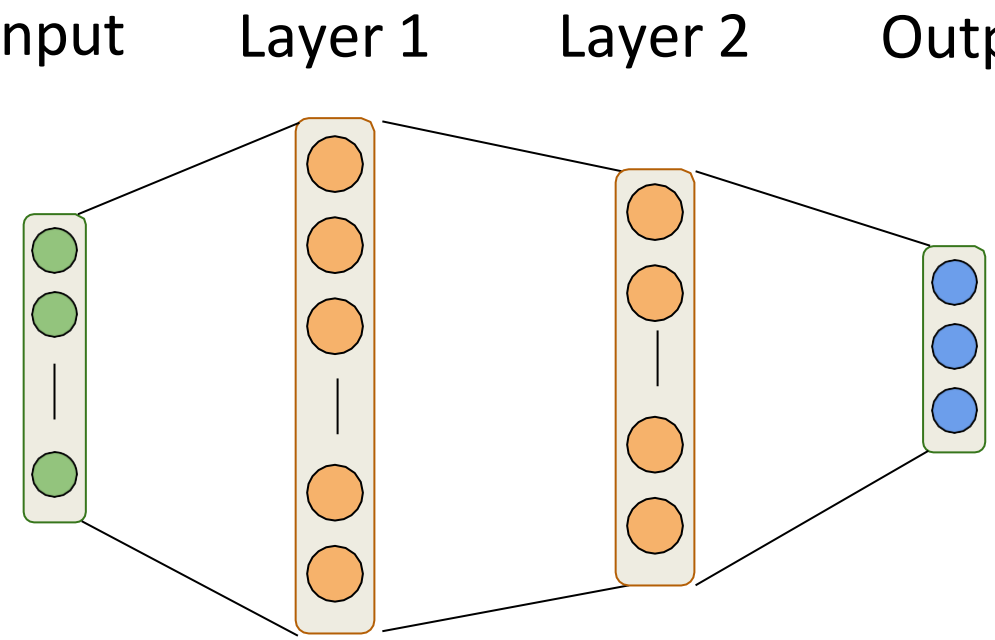
Neural Network: Optimization



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

Cross Entropy Loss

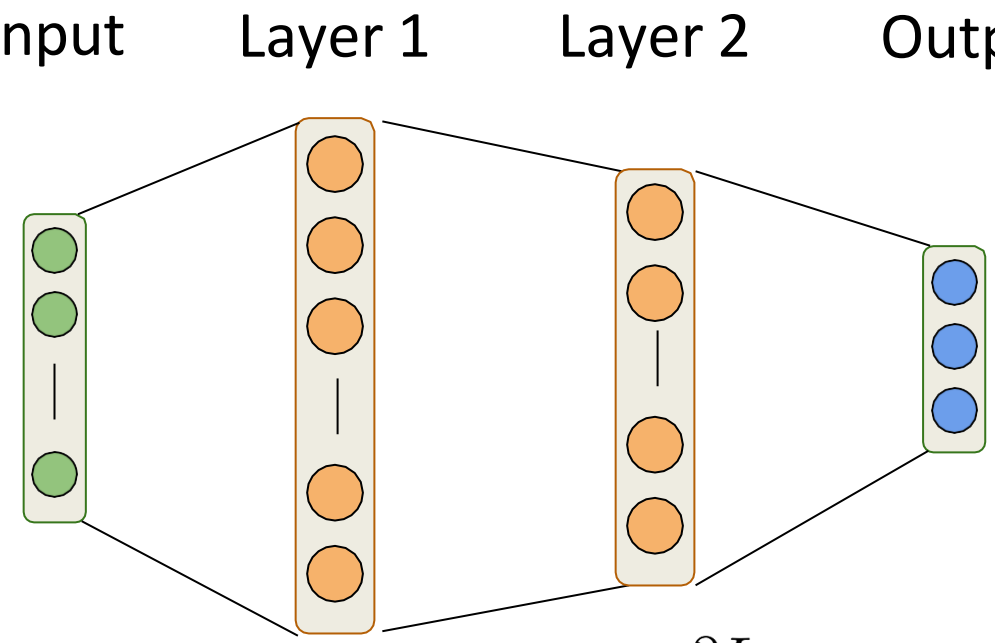
Neural Network: Optimization



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

Optimization

Neural Network: Optimization

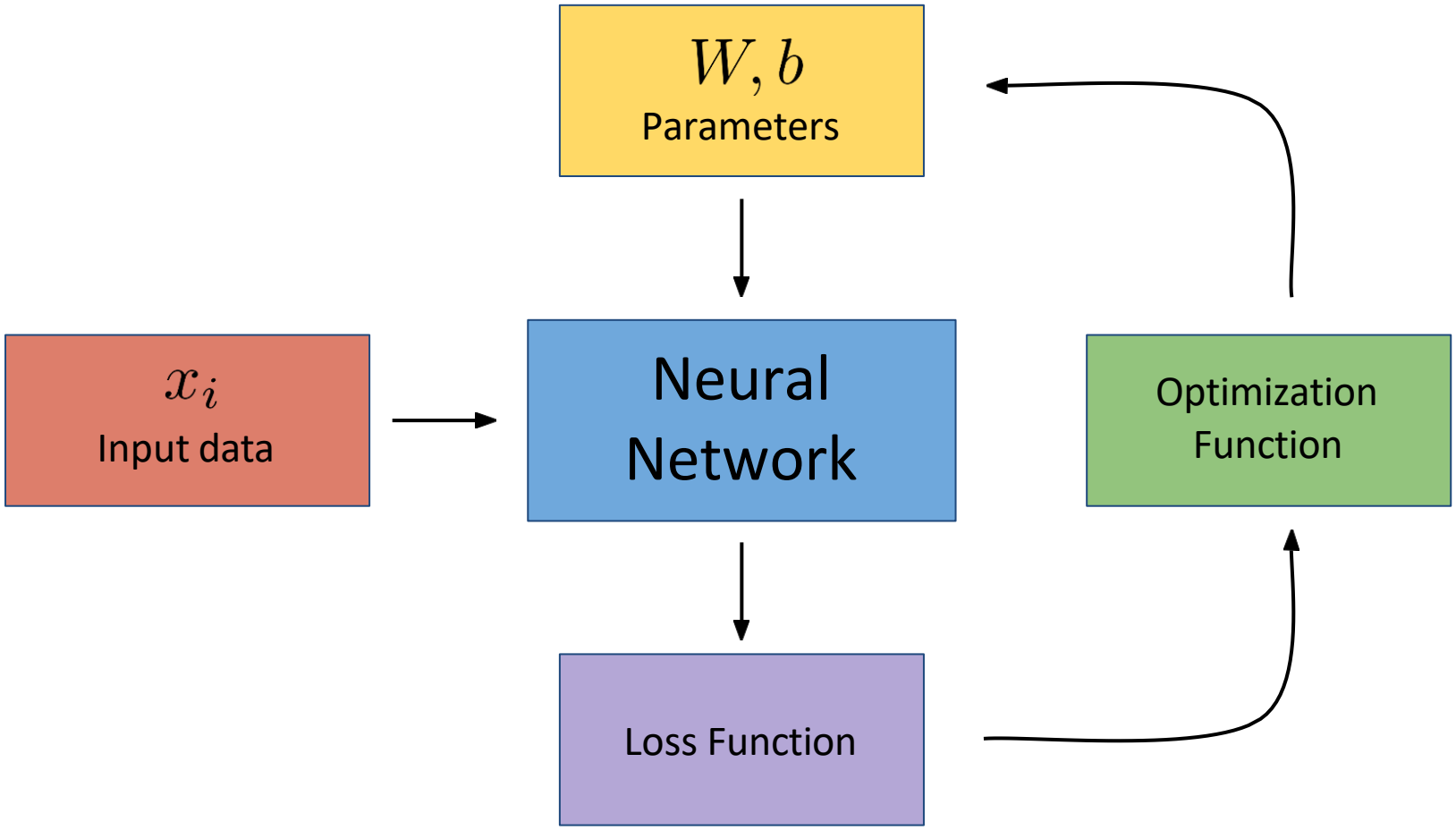


$$W_i = W_i - \eta \cdot \frac{\partial L}{\partial W_i}$$

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

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()
```

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))
model.add(Softmax())

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

Output 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()
```

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

Neural Network

Model Learning Curve

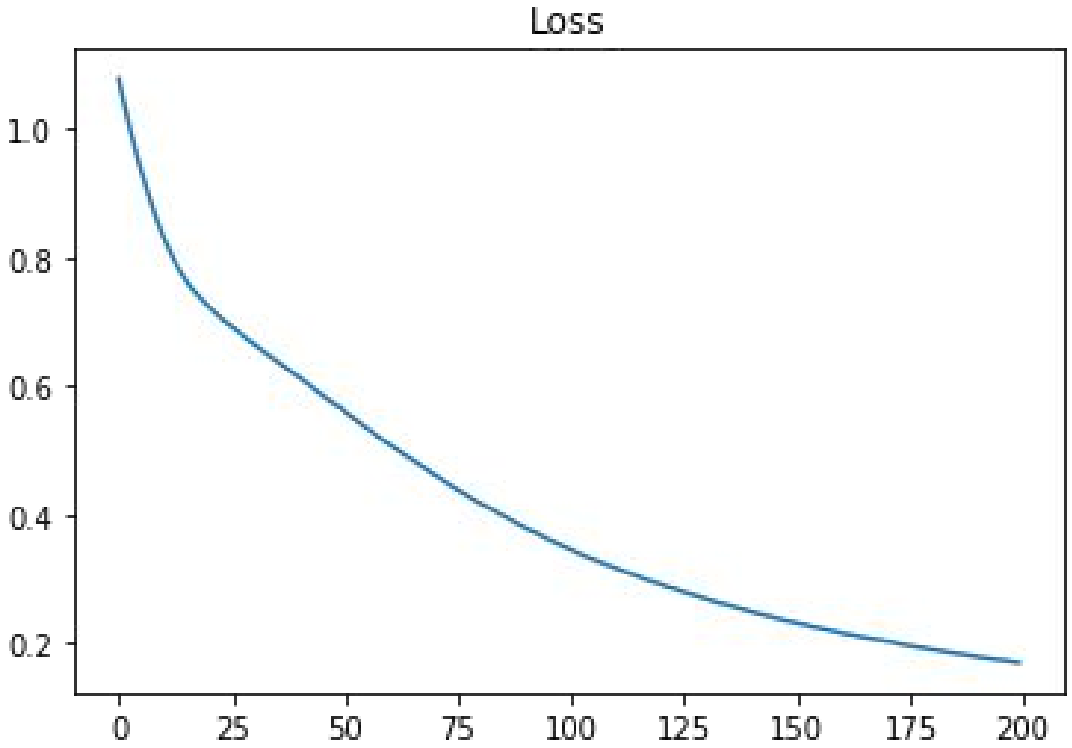
```
history = model.fit(X, y_probs, epochs=200, verbose=False)
plt.figure()
plt.plot(history.history['acc'])
plt.title("Accuracy")
plt.ylim((0.0, 1.01))

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

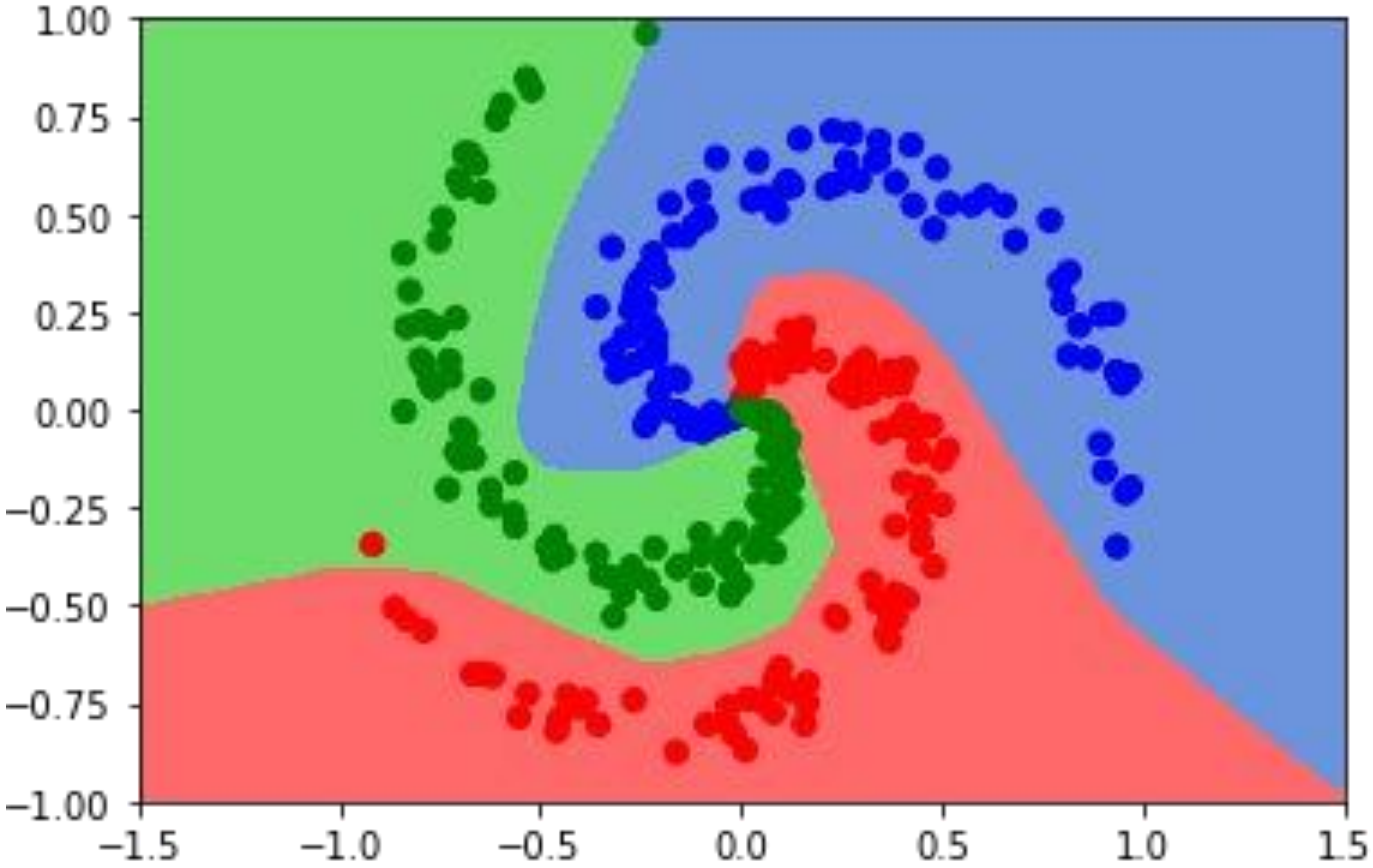
Model Learning Curve



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



Neural Network

Is the learning because of hidden layer or because of non-linearity added by activation functions?

Exercise:

1) Remove ReLU but keep one hidden layer and report the score
2) See the effect of learning rate (Hint: modify the code so that you use an explicitly initialize *Adam* optimizer object)

Neural Network

Some terminologies:

- Fully connected neural network
- Feed-forward neural network

Neural Network Language Model

Language Model

You shall know a word by the company it keeps

—Firth, J. R. 1957:11

Language Model

Fill in the blank:

... a _____ ...

car	cars	water	cat

Language Model

Fill in the blank:

… a _____ …

car and *cat* both work

car	cars	water	cat

Language Model

Fill in the blank:

… a _____ …

car and *cat* both work

car	cars	water	cat

John is driving a _____ …

Language Model

Fill in the blank:

... a _____ ...

car and *cat* both work

car	cars	water	cat

John is driving a _____ ...

only *car* works here

Language Model

Fill in the blank:

... a _____ ...

car and *cat* both work

car	cars	water	cat

John is driving a _____ ...

Similarly, machines use the context to predict the next words

Language Model

You chose “driving a car” because you’ve seen that phrase more frequently

“driving a cat” is not a common phrase

Language Model

Fill in the blank:

This _____ is going at 100 km/hours

car

bicycle

Car at 100km/hours is more probable than a bicycle

Language Model

Language model defines
“how probable a sentence is”

Language Model

Let's look at the example again

How probable is:

John is driving a car vs. John is driving a cat

In other words, what is the probability to predict cat or car given the context "John is driving a"

Language Model

Q: Can we see language modeling as a classification problem?

Language Model

Q: Can we see language modeling as a classification problem?

A: Yes! We are just predicting which word (“class”) is coming next.

Language Model

Q: Can we see language modeling as a classification problem?

A: Yes! We are just predicting which word ("class") is coming next.

$$p(\texttt{<s>}\ \text{Dan likes ham}\ \texttt{</s>}) =$$

Language Model

Q: Can we see language modeling as a classification problem?

A: Yes! We are just predicting which word ("class") is coming next.

$$p(\texttt{<s>}\ \text{Dan likes ham}\ \texttt{</s>}) = p(\texttt{Dan}|\texttt{<s>})$$

Predict **Dan** given **<s>**

Language Model

Q: Can we see language modeling as a classification problem?

A: Yes! We are just predicting which word ("class") is coming next.

$$p(\texttt{<s>}\ \text{Dan likes ham}\ \texttt{</s>}) = p(\texttt{Dan}|\texttt{<s>})$$

Predict **Dan** given **<s>**

What is the probability of **Dan** given **<s>**?

Language Model

Q: Can we see language modeling as a classification problem?

A: Yes! We are just predicting which word ("class") is coming next.

$$p(\texttt{<s>}\ \text{Dan likes ham}\ \texttt{</s>}) = p(\texttt{Dan}|\texttt{<s>}) \cdot p(\texttt{likes}|\texttt{Dan})$$

Predict **likes** given **Dan**

Language Model

Q: Can we see language modeling as a classification problem?

A: Yes! We are just predicting which word ("class") is coming next.

$$p(\texttt{<s>}\ \text{Dan likes ham}\ \texttt{</s>}) = p(\texttt{Dan}|\texttt{<s>}) \cdot p(\texttt{likes}|\texttt{Dan}) \cdot p(\texttt{ham}|\texttt{likes})$$

Predict **ham** given **likes**

Language Model

Q: Can we see language modeling as a classification problem?

A: Yes! We are just predicting which word ("class") is coming next.

$$p(\texttt{<s>}\ \text{Dan likes ham}\ \texttt{</s>}) = p(\texttt{Dan}|\texttt{<s>}) \cdot p(\texttt{likes}|\texttt{Dan}) \cdot p(\texttt{ham}|\texttt{likes}) \cdot p(\texttt{</s>}|\texttt{ham})$$

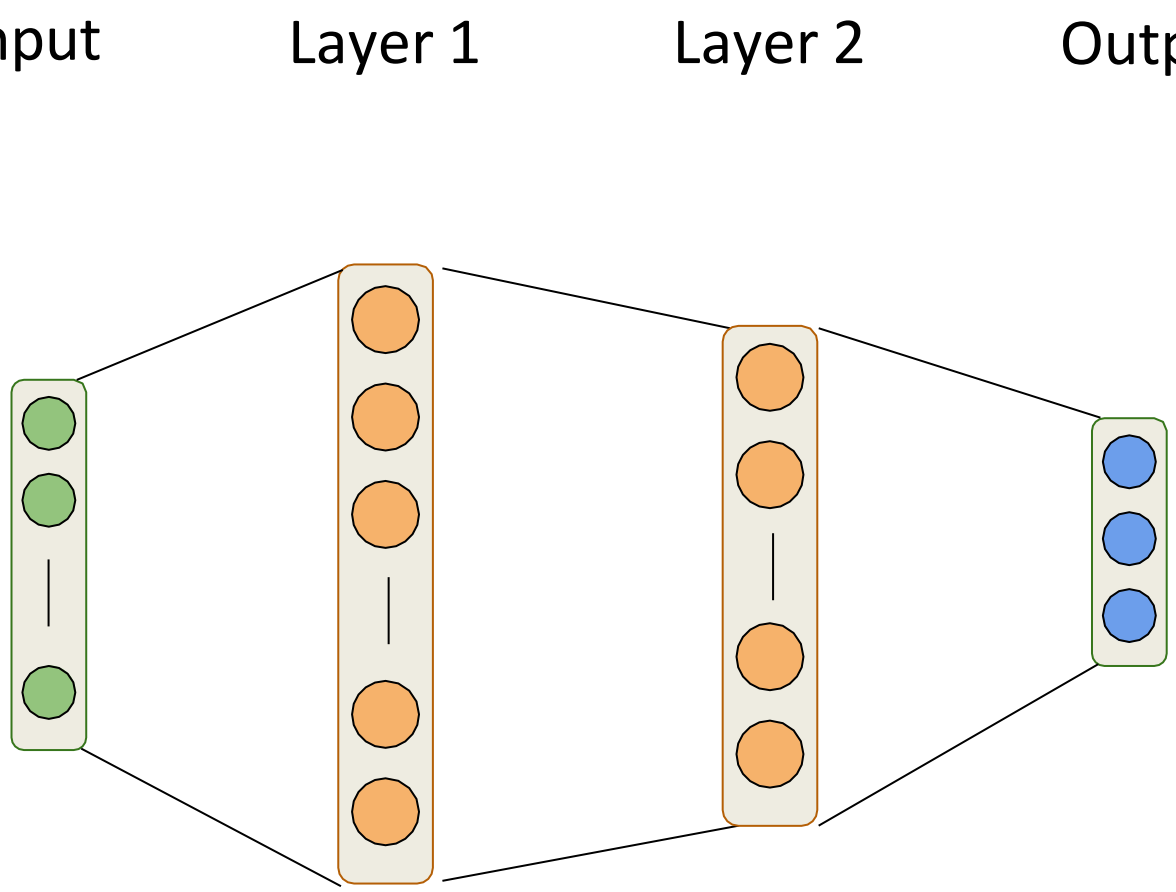
Language Model

Words represent classes that we want to predict!

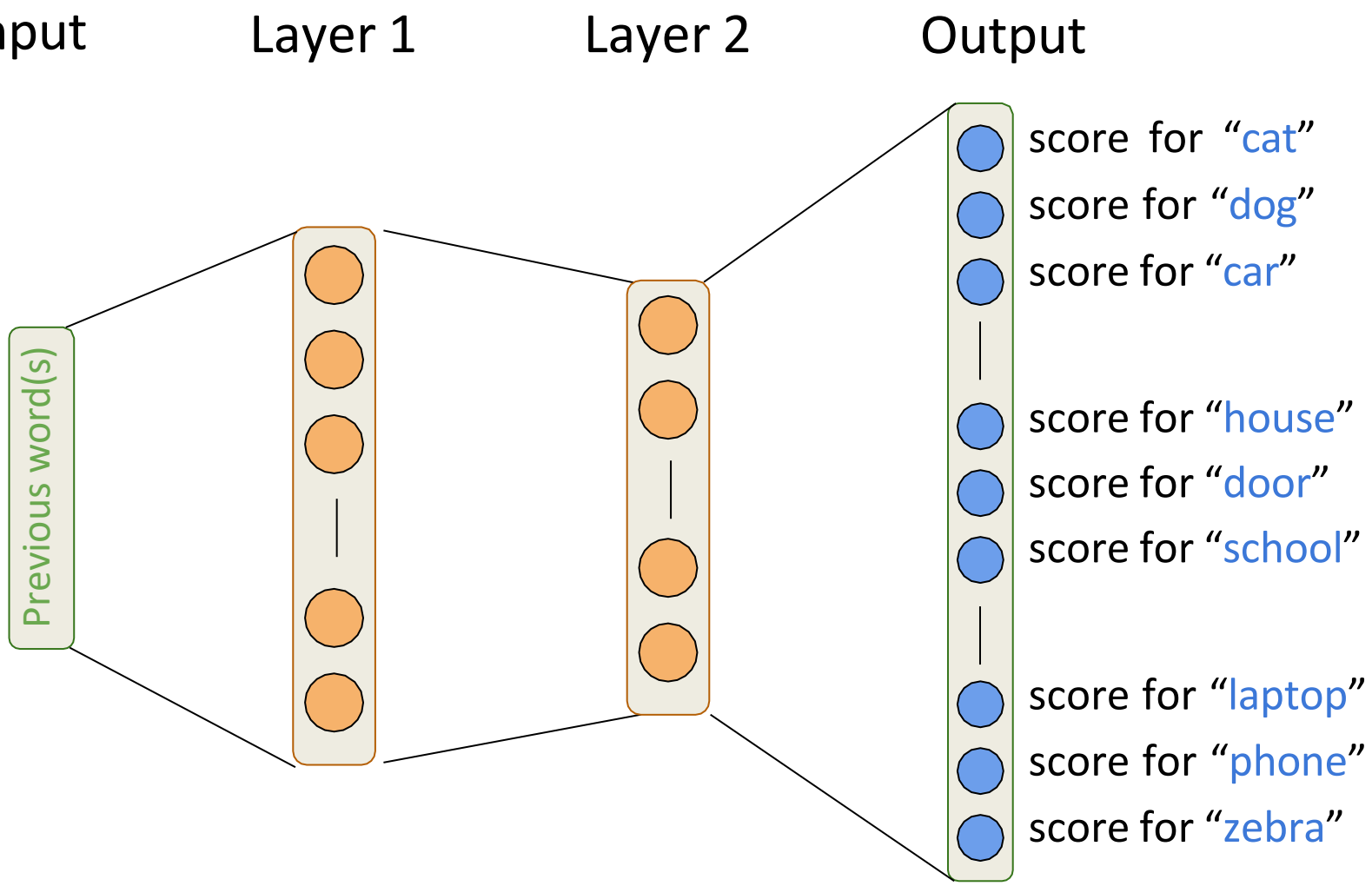
Input to the classifier: previous words i.e. context

Output: probability distribution over all possible words, i.e. our vocabulary

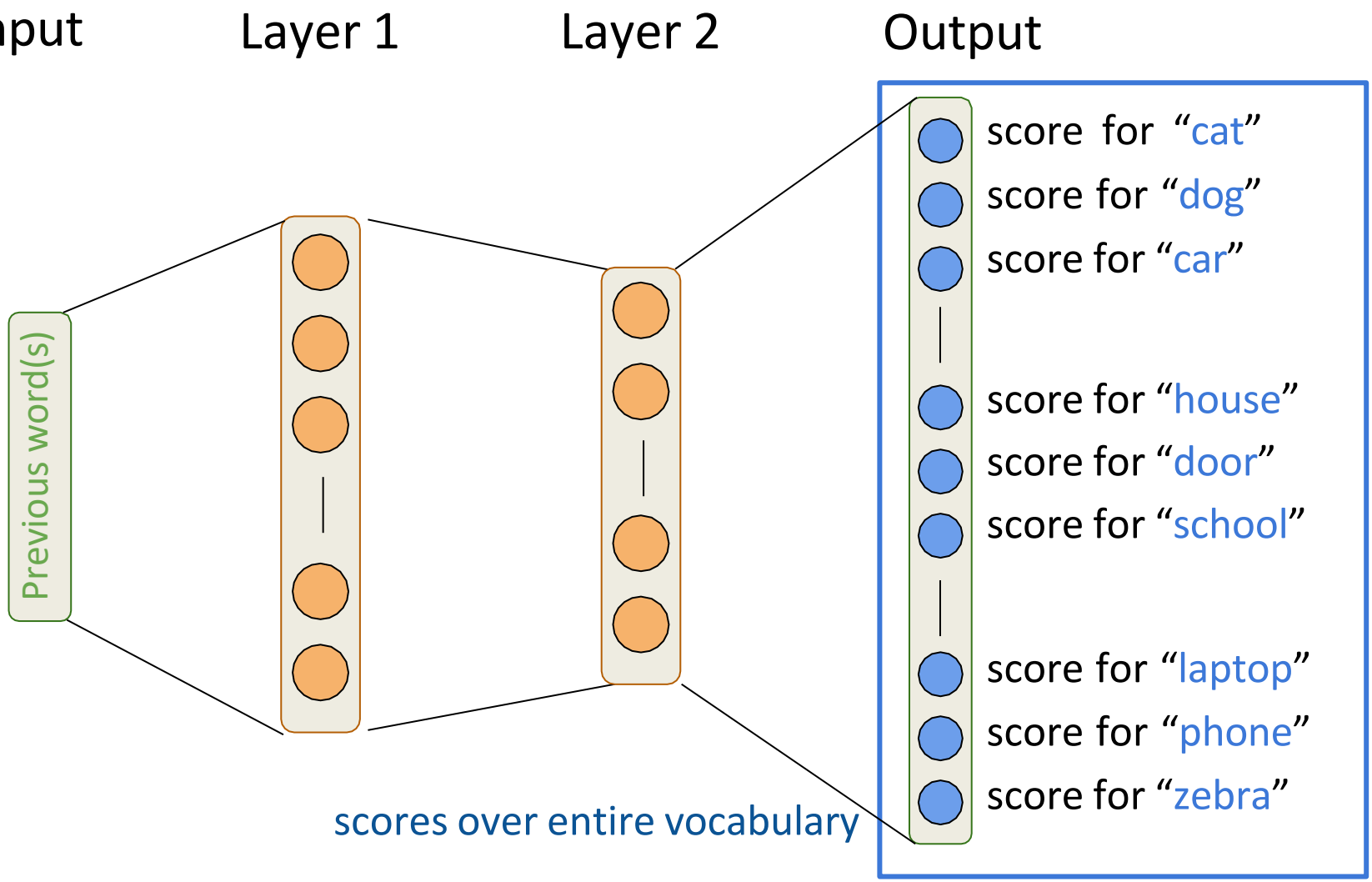
Neural Network Language Model



Neural Network Language Model



Neural Network Language Model



Input Representation

Input

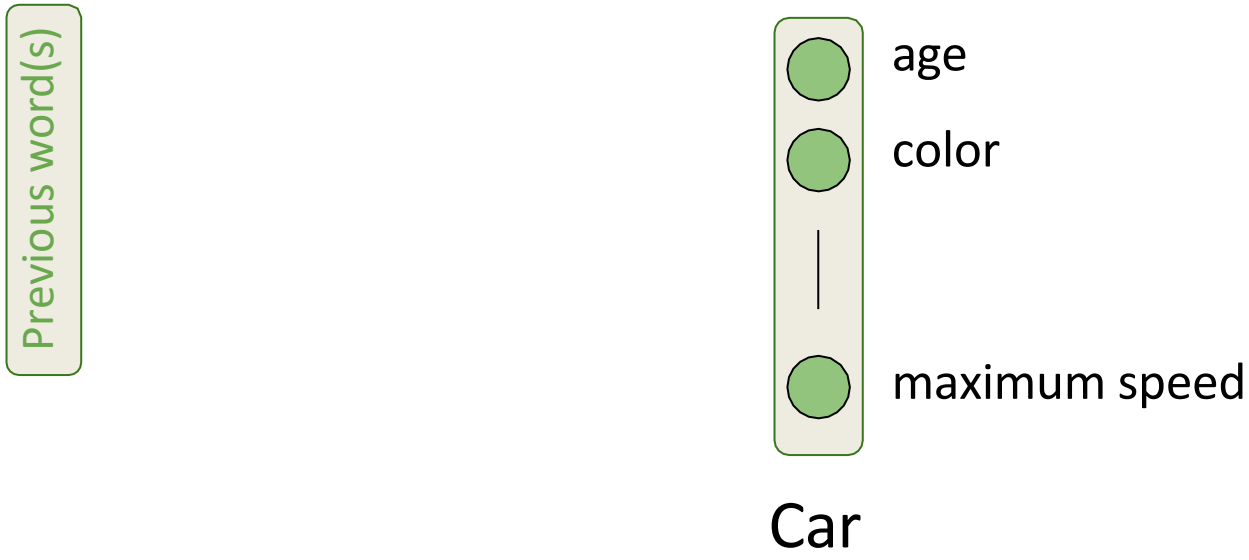
Previously we've used a vector as input, where each element of the vector represented some "feature" of the input

Previous word(s)

Input Representation

Input

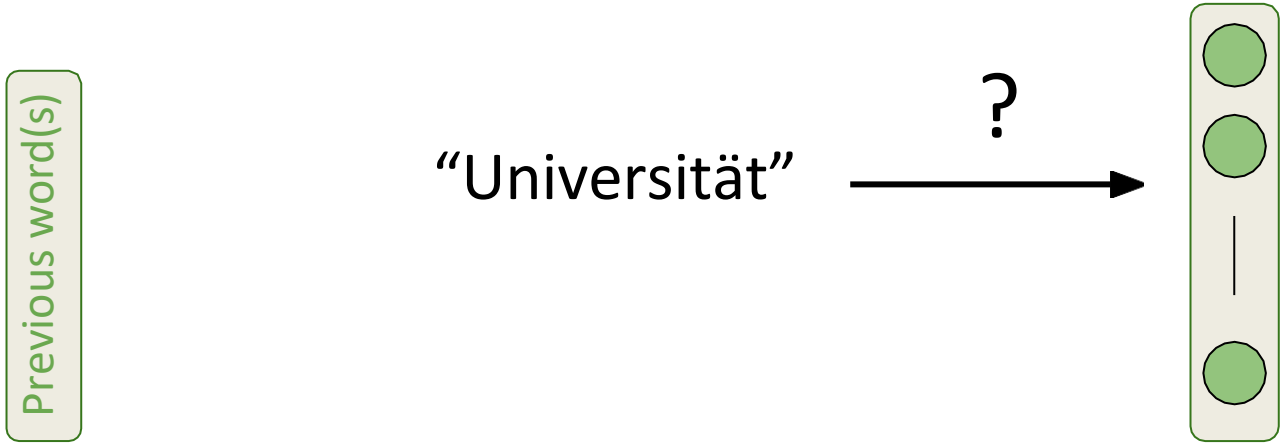
Previously we've used a vector as input, where each element of the vector represented some "feature" of the input



Input Representation

Input

Can we represent a word as a feature vector?



One Hot Vector Representation

- Every word can be represented as a ***one hot vector***

One Hot Vector Representation

- Every word can be represented as a ***one hot vector***
- Suppose the total number of unique words in the corpus is 10,000

One Hot Vector Representation

- Every word can be represented as a ***one hot vector***
- Suppose the total number of unique words in the corpus is 10,000
- Assign each word a unique index:

```
Universität: 1
cat: 2
house: 3
car: 4
   ⋮
apple: 10,000
```

One Hot Vector Representation

- Every word can be represented as a ***one hot vector***
- Suppose the total number of unique words in the corpus is 10,000
- Assign each word a unique index:

```
Universität: 1
cat: 2
house: 3
car: 4
   ⋮
apple: 10,000
```

Dictionary

$$cat = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

One-hot representation

One Hot Vector Representation

- Every word can be represented as a ***one hot vector***
- Suppose the total number of unique words in the corpus is 10,000
- Assign each word a unique index:

Only index that represents the input word will be one

$$cat = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

One-hot representation

One Hot Vector Representation

- Every word can be represented as a ***one hot vector***
- Suppose the total number of unique words in the corpus is 10,000
- Assign each word a unique index:

Only index that represents the input word will be one

$$cat = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \qquad car = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

One-hot representation

One Hot Vector Representation

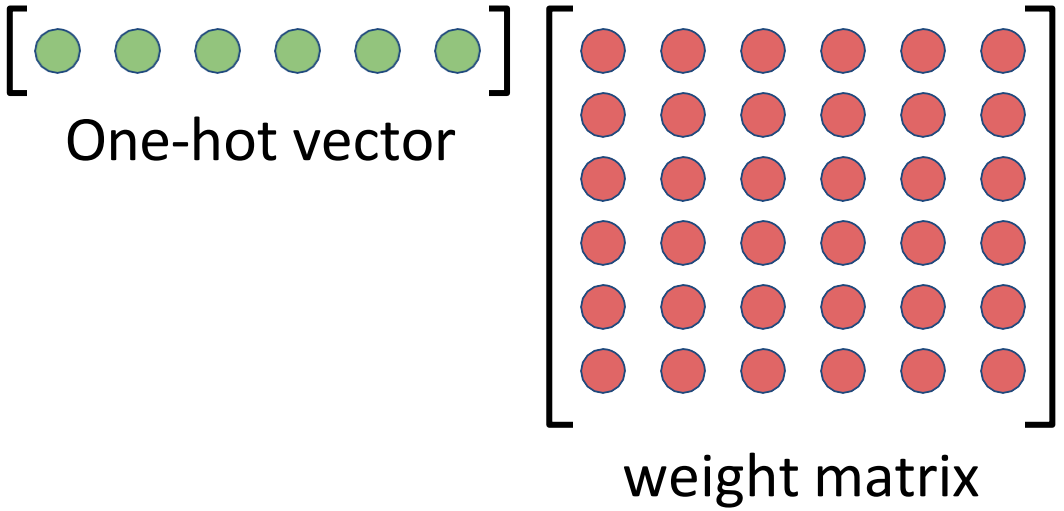
- Every word can be represented as a ***one hot vector***
- Suppose the total number of unique words in the corpus is 10,000
- Assign each word a unique index:

Vector size will be the size of the vocabulary, i.e. 10,000 in this case

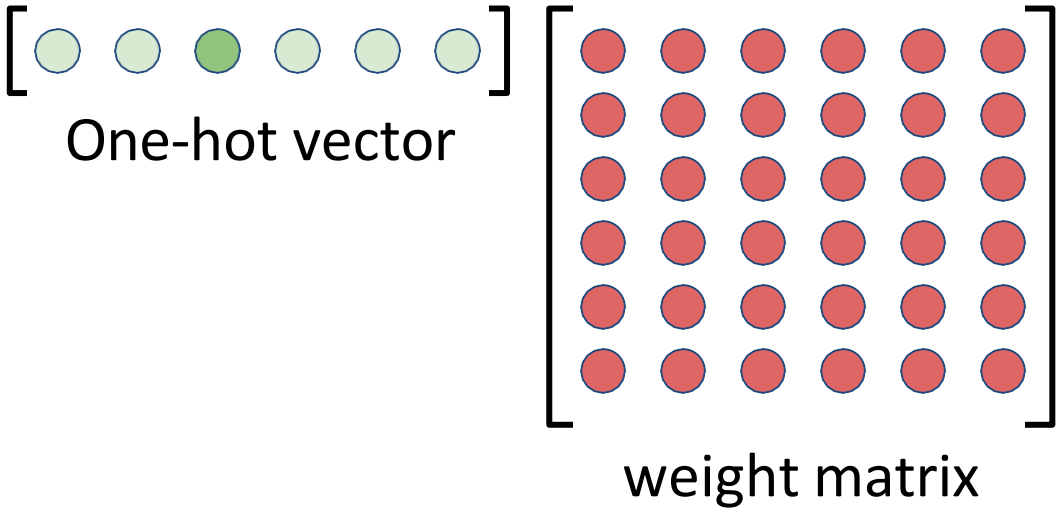
$$cat = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \qquad car = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

One-hot representation

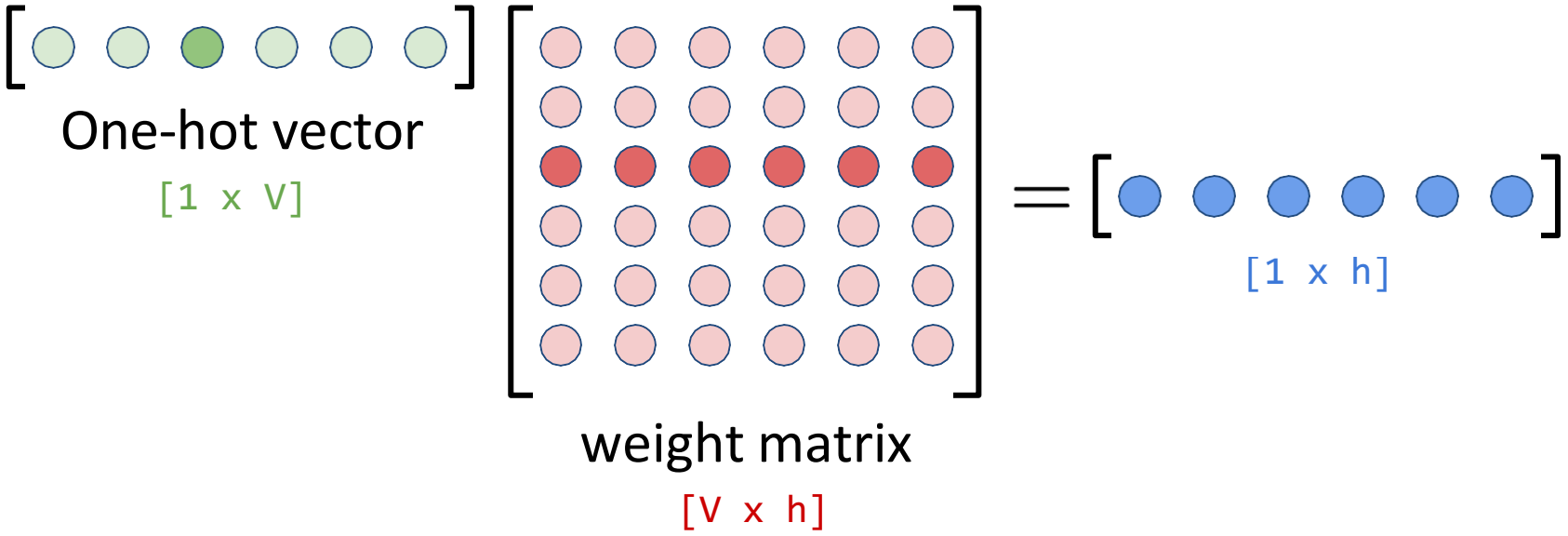
One Hot Vector Representation



One Hot Vector Representation



One Hot Vector Representation



One-hot vector will “turn on” one row of weights

Higher ngram Vector Representation

- What about representing multiple words?

Bag of words approach

Higher ngram Vector Representation

- What about representing multiple words?

Bag of words approach

Bigram: indices of the *two previous words* are 1 in the vector

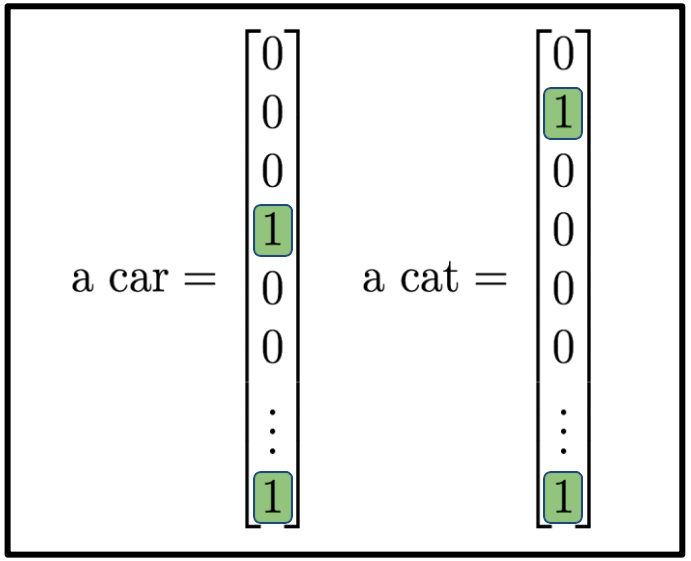
$$\text{a car} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} \qquad \text{a cat} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}$$

Higher ngram Vector Representation

- What about representing multiple words?

Bag of words approach

Bigram: indices of the *two previous words* are 1 in the vector

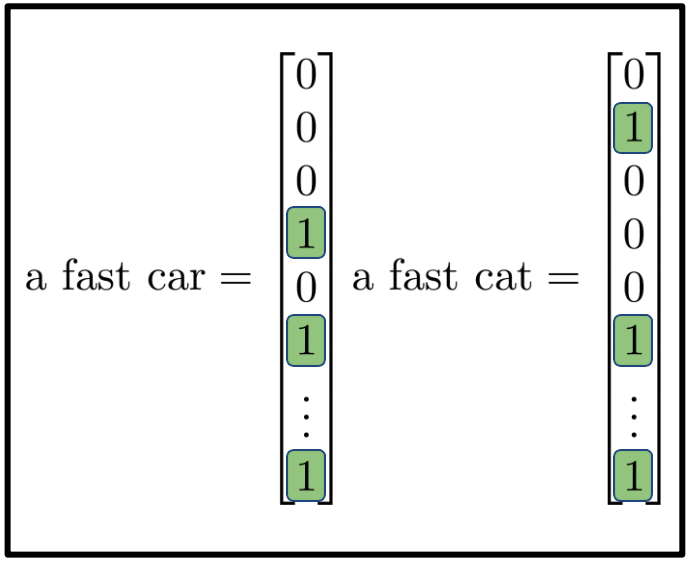


Higher ngram Vector Representation

- What about representing multiple words?

Bag of words approach

Trigram: indices of the *three previous words* are 1 in the vector



Higher ngram Vector Representation

- What about representing multiple words?

Context-aware approach

In the **bag of words** approach, order information is lost!

Higher ngram Vector Representation

- What about representing multiple words?

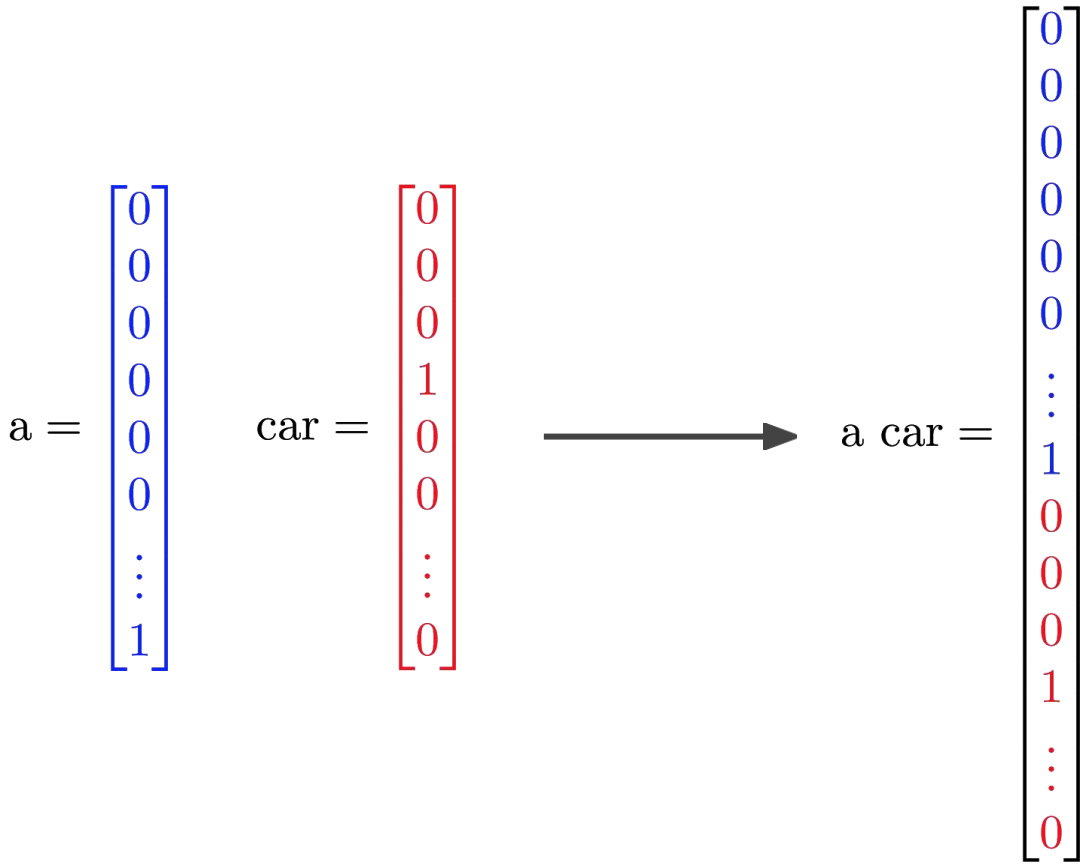
Context-aware approach

In the **bag of words** approach, order information is lost!

Solution: for *N* words, concatenate one-hot vectors for each of the words in the correct order

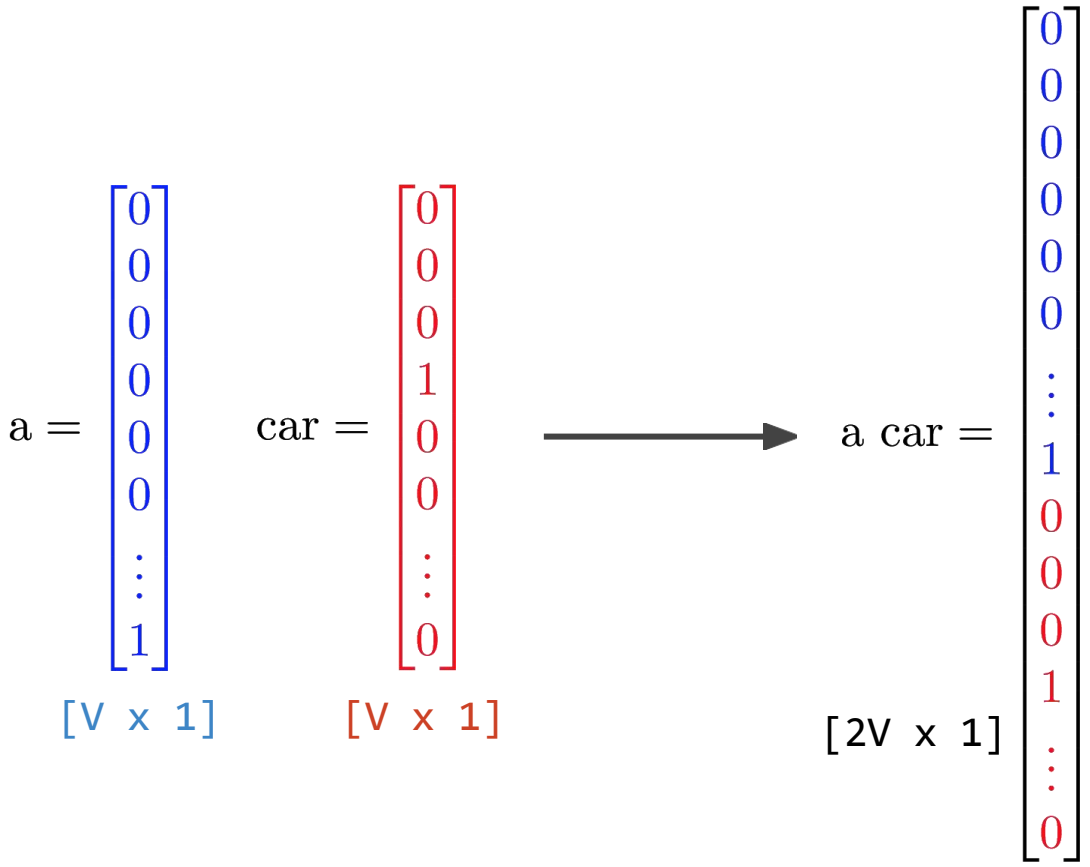
Higher ngram Vector Representation

Context-aware approach



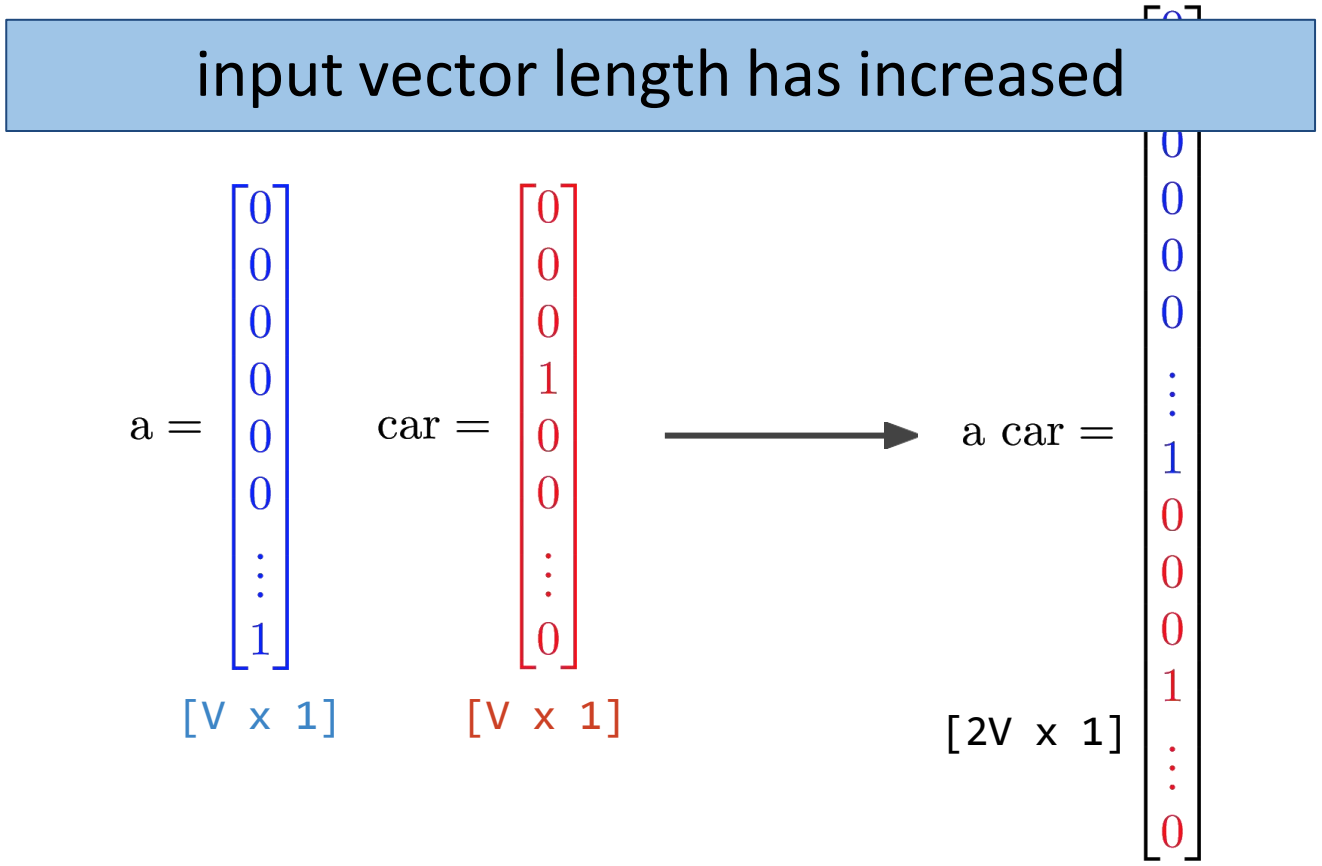
Higher ngram Vector Representation

Context-aware approach



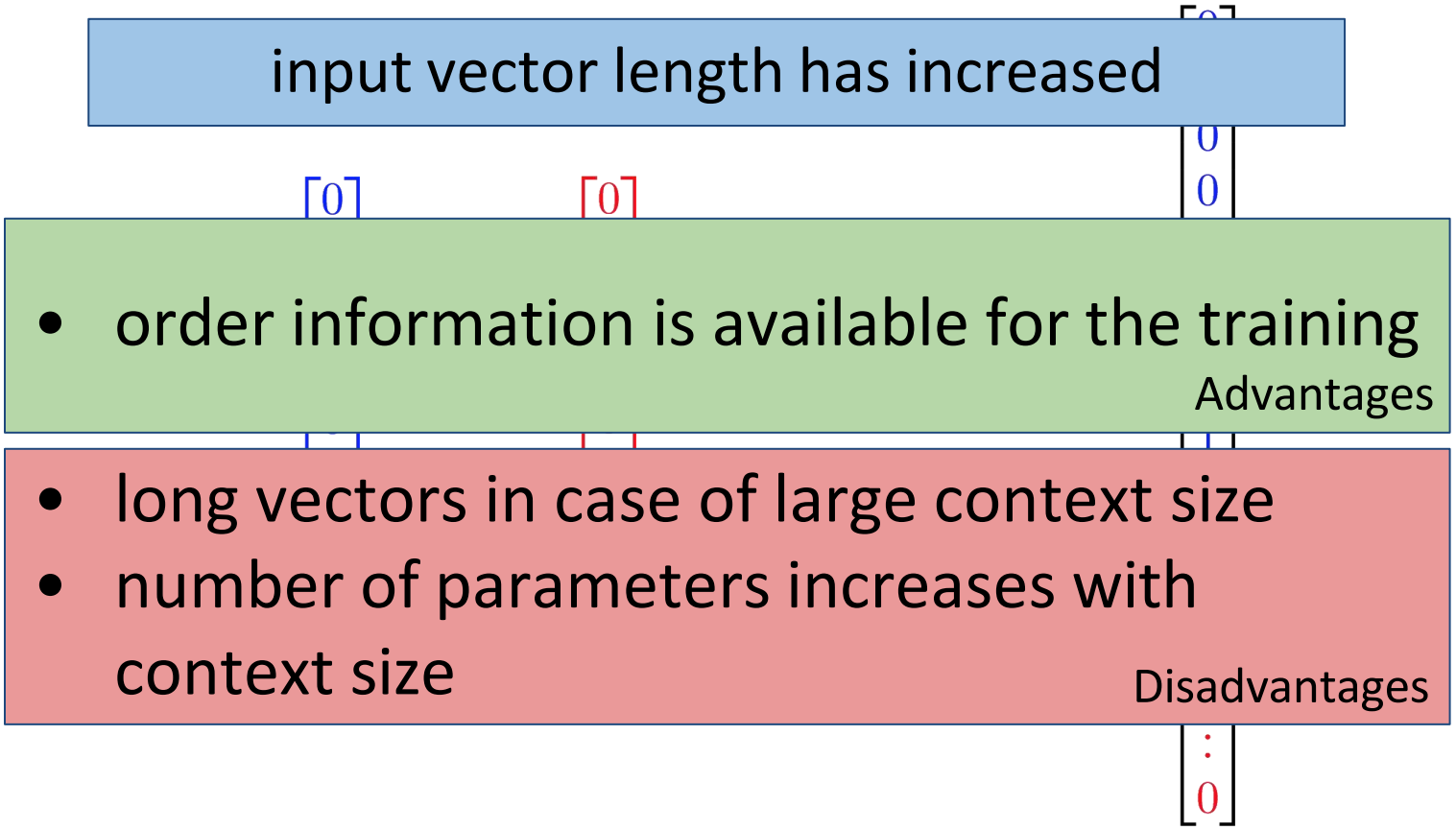
Higher ngram Vector Representation

Context-aware approach



Higher ngram Vector Representation

Context-aware approach



Higher ngram Vector Representation

- Bag of words vs. context-aware approach?
  - Given the disadvantages of the context-aware approach, Bag of words is more commonly used
  - Works well in practice

Input Representation

Generally, the size of the vocabulary is very large

- Results in very large one-hot vectors!

Input Representation

Generally, the size of the vocabulary is very large

- Results in very large one-hot vectors!

Some tricks to reduce vocabulary size:

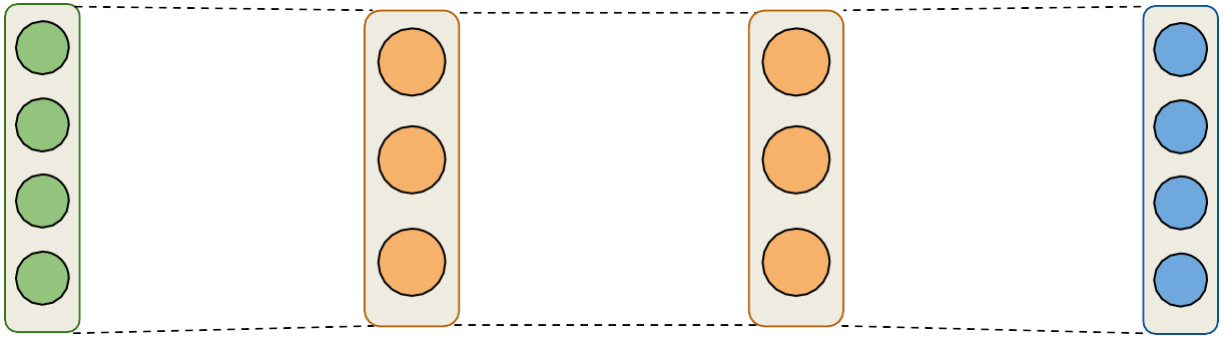
1) Take most frequent top words. For example, consider only 10,000 most frequent words and map the rest to a unique token <UNK>
2) Cluster words
   a) based on context
   b) based on linguistic properties

Neural Network Language Model

Let us look at a complete example:

Vocabulary: {“how”, “you”, “hello”, “are”}

Network Architecture: 2 hidden layers of size 3 each

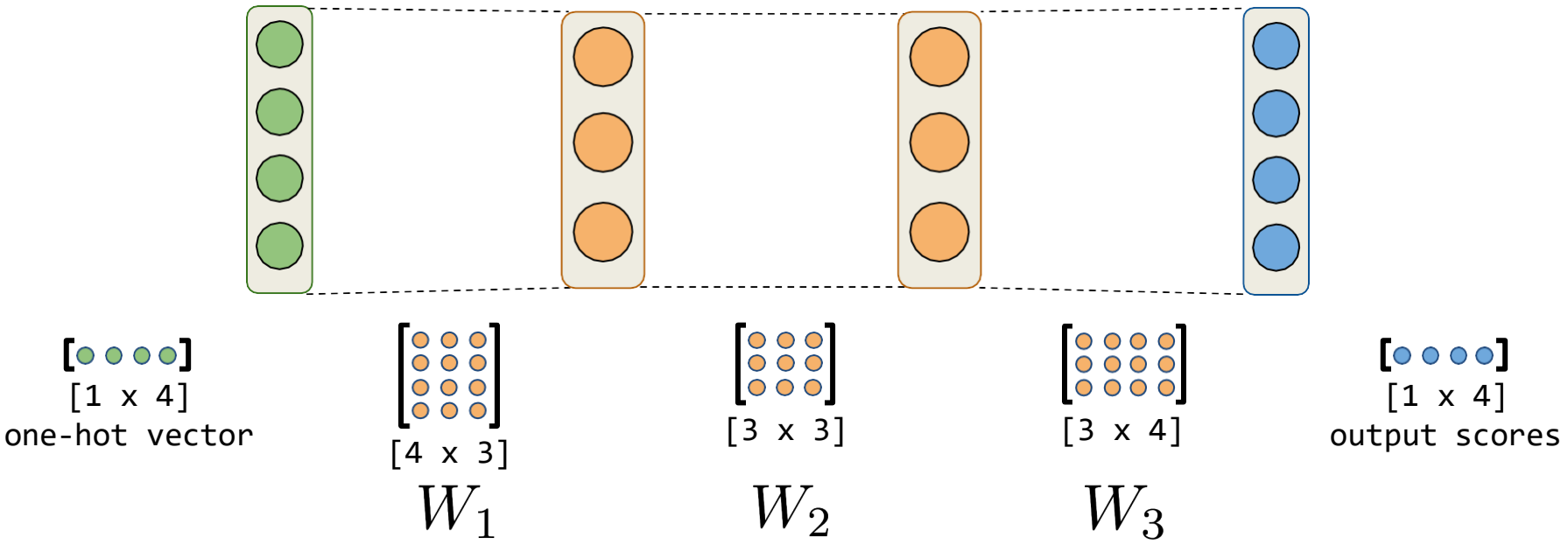


Neural Network Language Model

Let us look at a complete example:

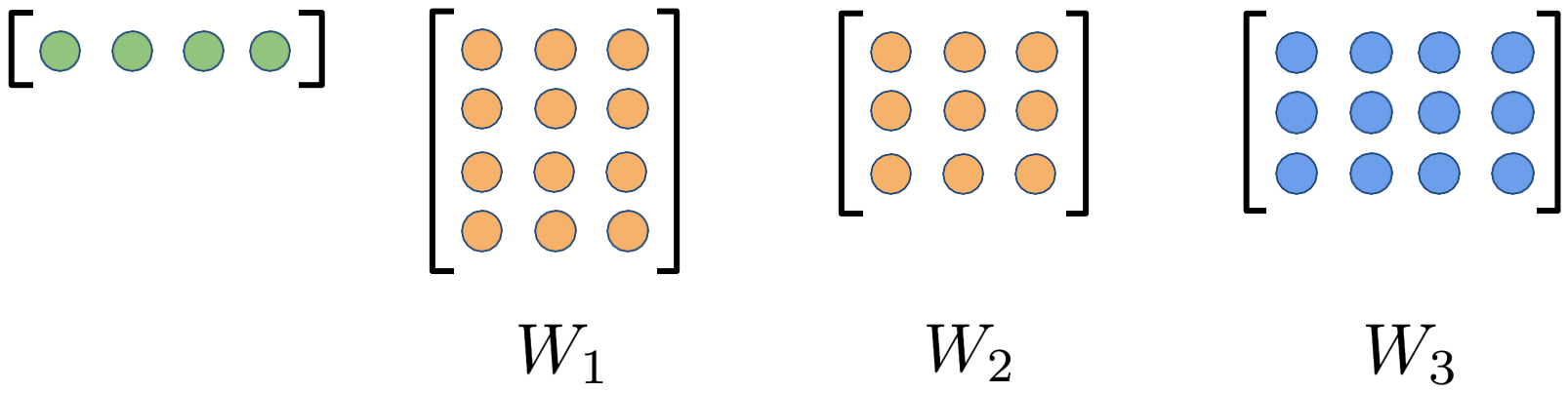
Vocabulary: {“how”, “you”, “hello”, “are”}

Network Architecture: 2 hidden layers of size 3 each



Neural Network Language Model

Vocabulary: {“how”, “you”, “hello”, “are”}



Neural Network Language Model

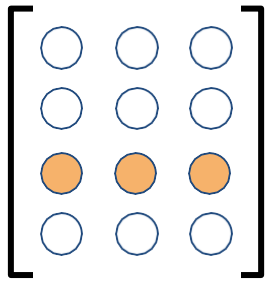
Vocabulary: {“how”, “you”, “hello”, “are”}

“hello”

Neural Network Language Model

Vocabulary: {“how”, “you”, “hello”, “are”}

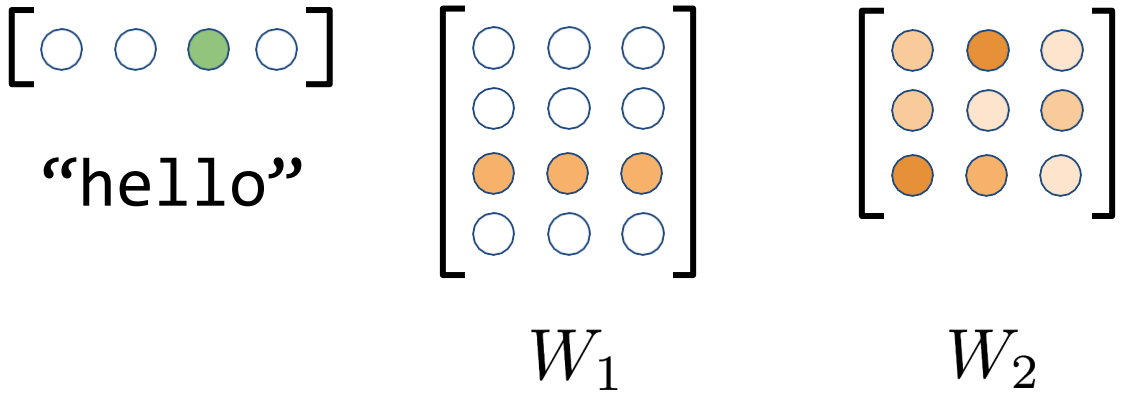
“hello”



W_1

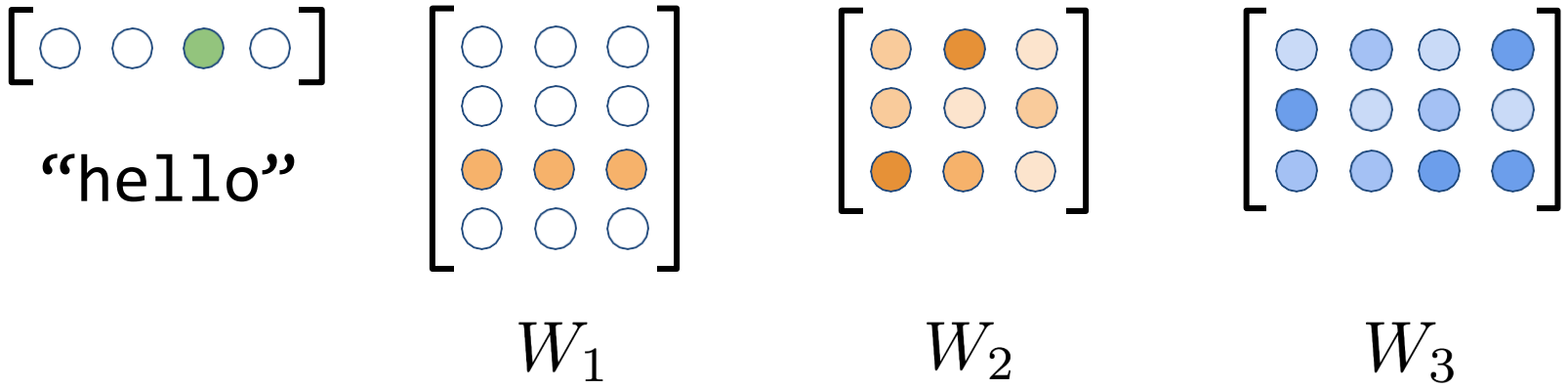
Neural Network Language Model

Vocabulary: {“how”, “you”, “hello”, “are”}



Neural Network Language Model

Vocabulary: {“how”, “you”, “hello”, “are”}

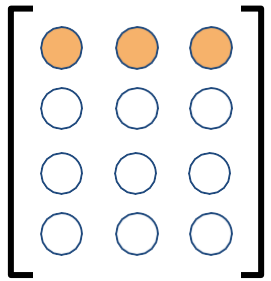


output scores
max: “how”

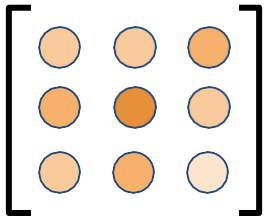
Neural Network Language Model

Vocabulary: {“how”, “you”, “hello”, “are”}

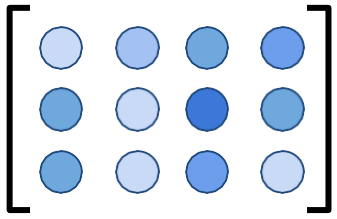
“how”



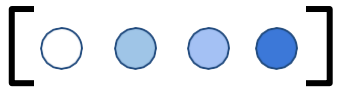
W_1



W_2



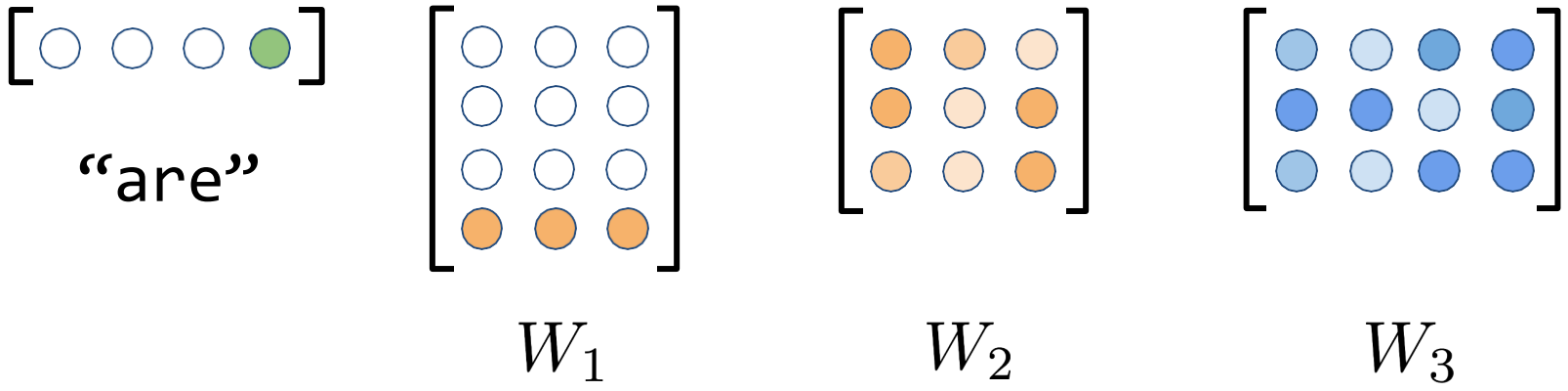
W_3



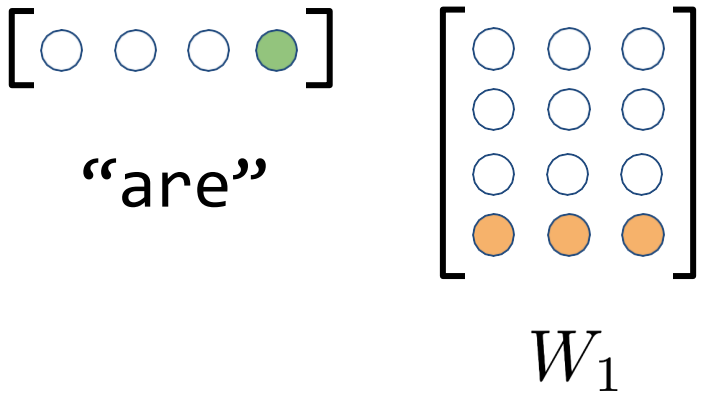
output scores
max: “are”

Neural Network Language Model

Vocabulary: {“how”, “you”, “hello”, “are”}

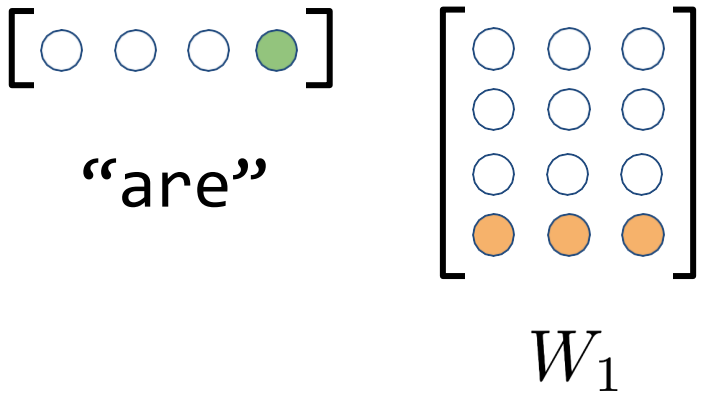


Neural Network Language Model



Each one-hot vector turns on one row in the weight matrix and results in `[1 x 3]` vector

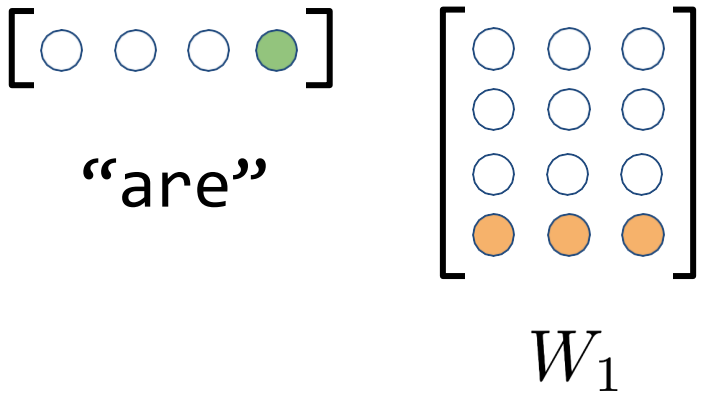
Neural Network Language Model



Each one-hot vector turns on one row in the weight matrix and results in `[1 x 3]` vector

Can we say that the `[1 x 3]` *vector represents the input word?*

Neural Network Language Model



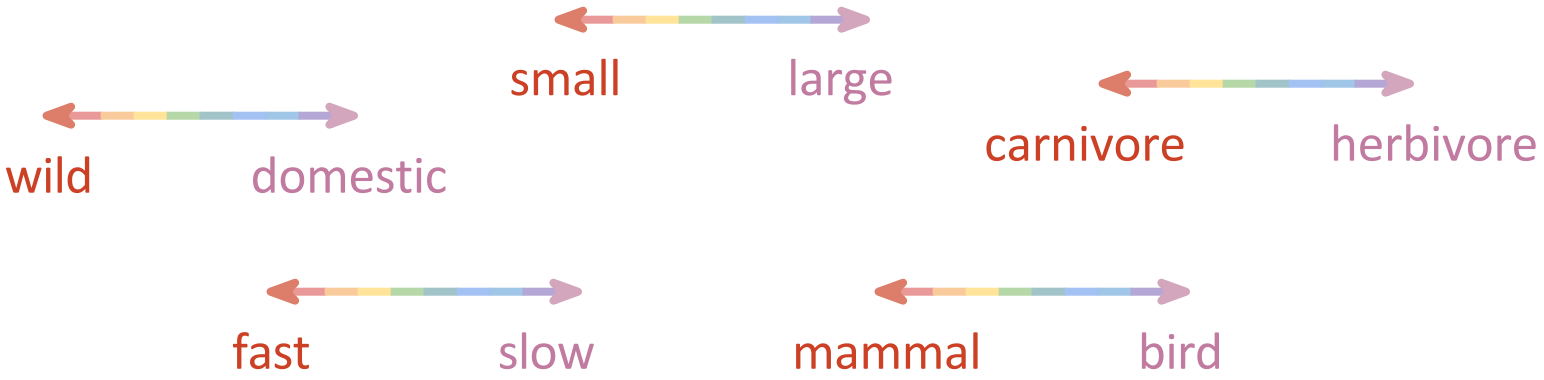
Each one-hot vector turns on one row in the weight matrix and results in `[1 x 3]` vector

Can we say that the `[1 x 3]` *vector represents the input word?* **Yes**

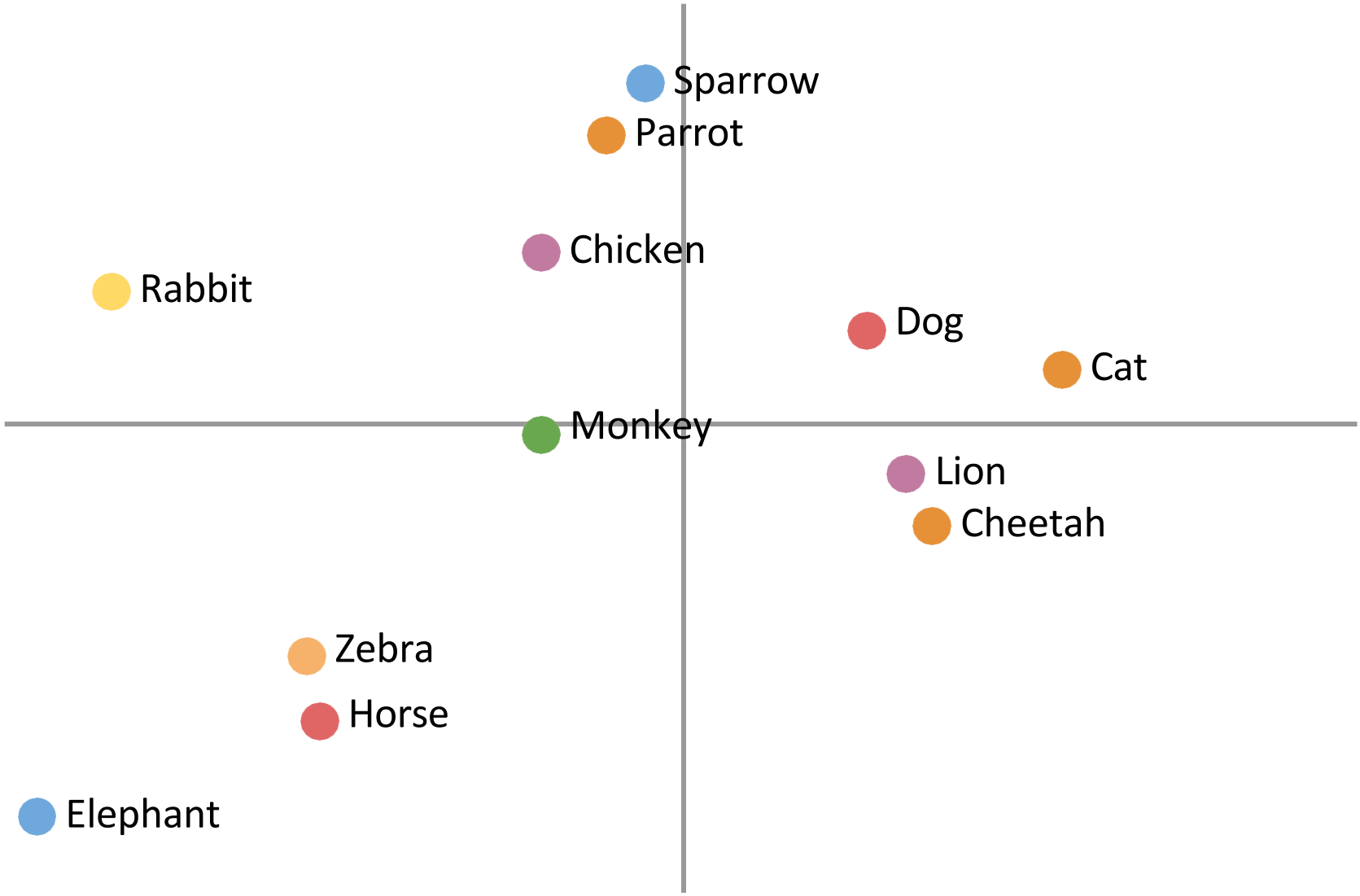
Exercise

Create a 2D vector space representation of the following words:

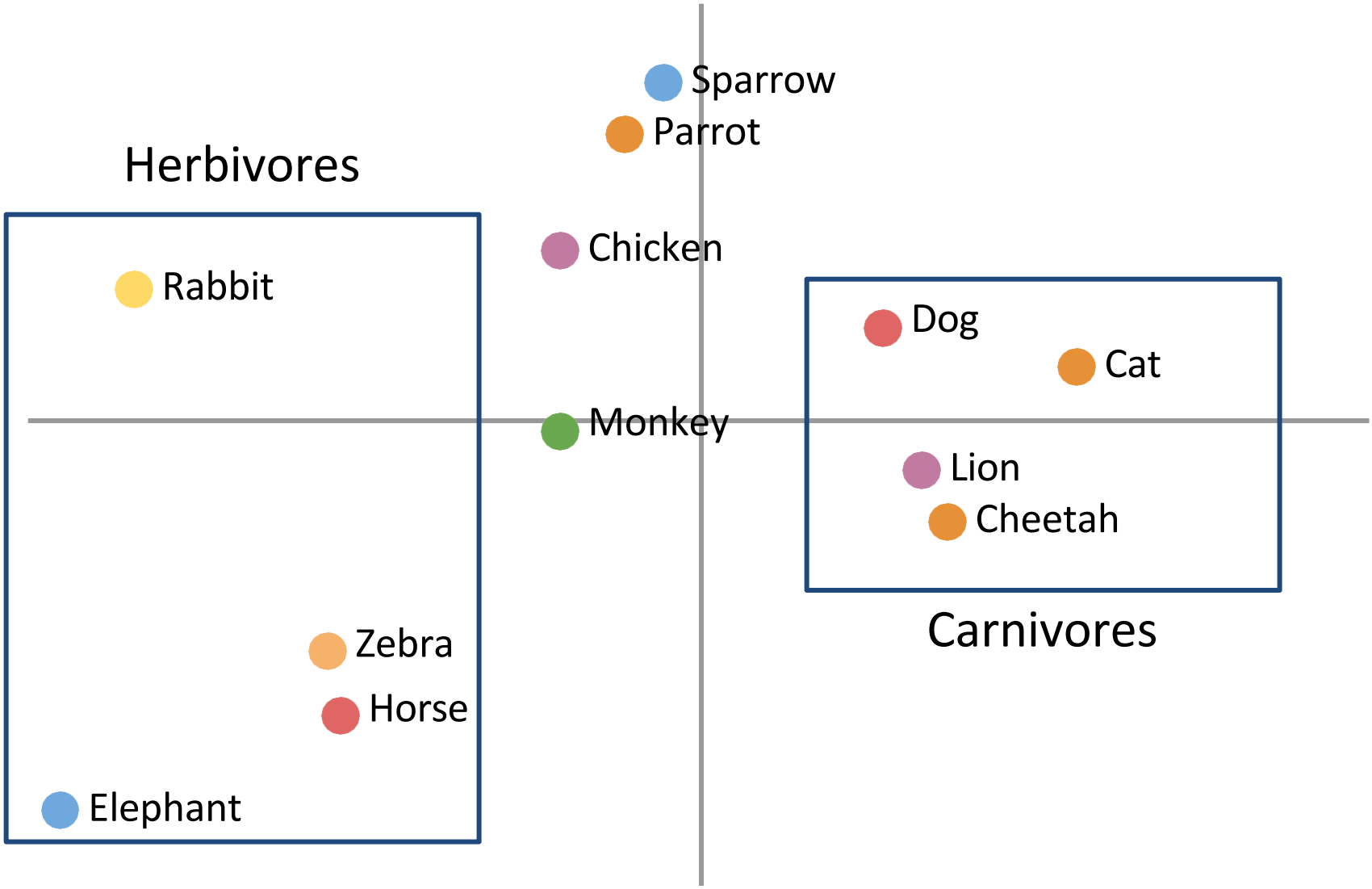
`dog, lion, cat, rabbit, horse, zebra, cheetah, parrot, sparrow, elephant, chicken, monkey`



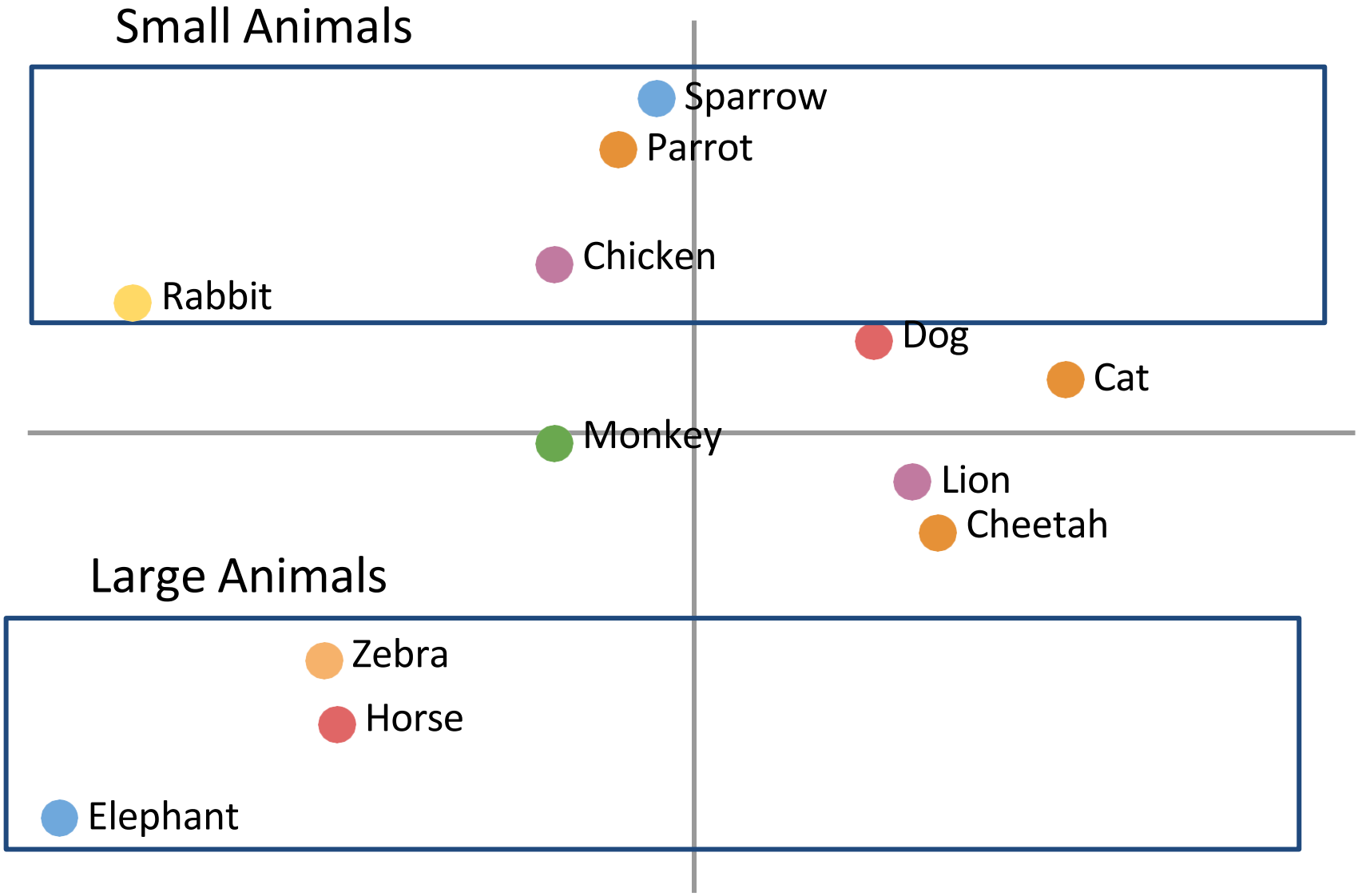
Exercise


Sparrow
Parrot
Chicken
Rabbit
Dog
Cat
Monkey
Lion
Cheetah
Zebra
Horse
Elephant

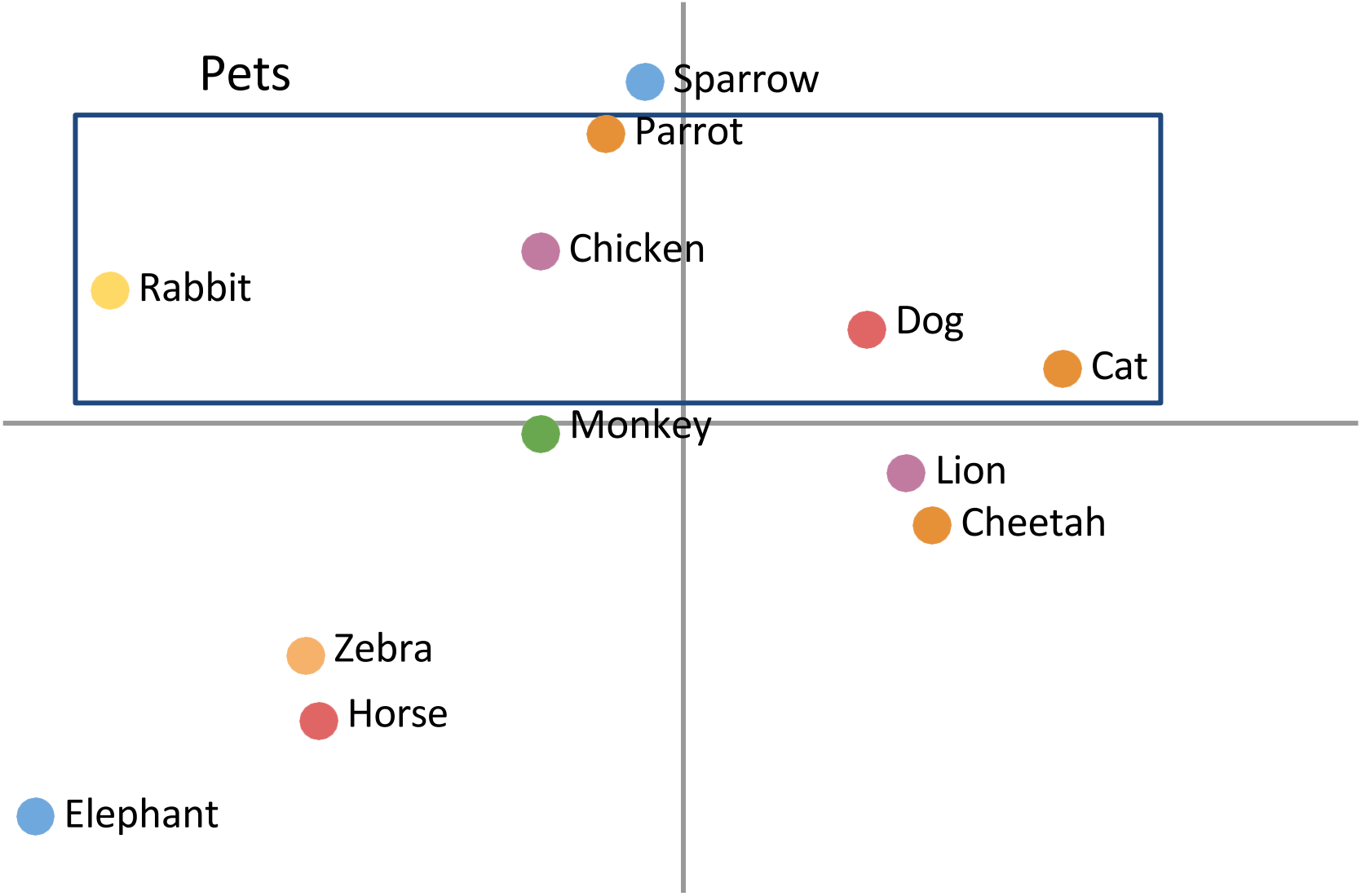
Exercise


Sparrow
Parrot
Herbivores
Chicken
Rabbit
Dog
Cat
Monkey
Lion
Cheetah
Zebra
Horse
Carnivores
Elephant

Exercise


Small Animals
Sparrow
Parrot
Chicken
Rabbit
Dog
Cat
Monkey
Lion
Cheetah
Large Animals
Zebra
Horse
Elephant

Exercise


Pets
Sparrow
Parrot
Chicken
Rabbit
Dog
Cat
Monkey
Lion
Cheetah
Zebra
Horse
Elephant

Word Embeddings

How did you decide which animals need to be closer?

How did you handle conflicts between animals that belong to multiple groups?

How does having this kind of vector space representation help us?

Word Embeddings

- In one-hot vector representation, a word is represented as one large ***sparse*** vector

only one element is 1 in the entire vector

vectors of different words do not give us any information about the potential relations between the words!

Word Embeddings

- In one-hot vector representation, a word is represented as one large ***sparse*** vector
- Instead, **word embeddings** are ***dense*** vectors in some vector space

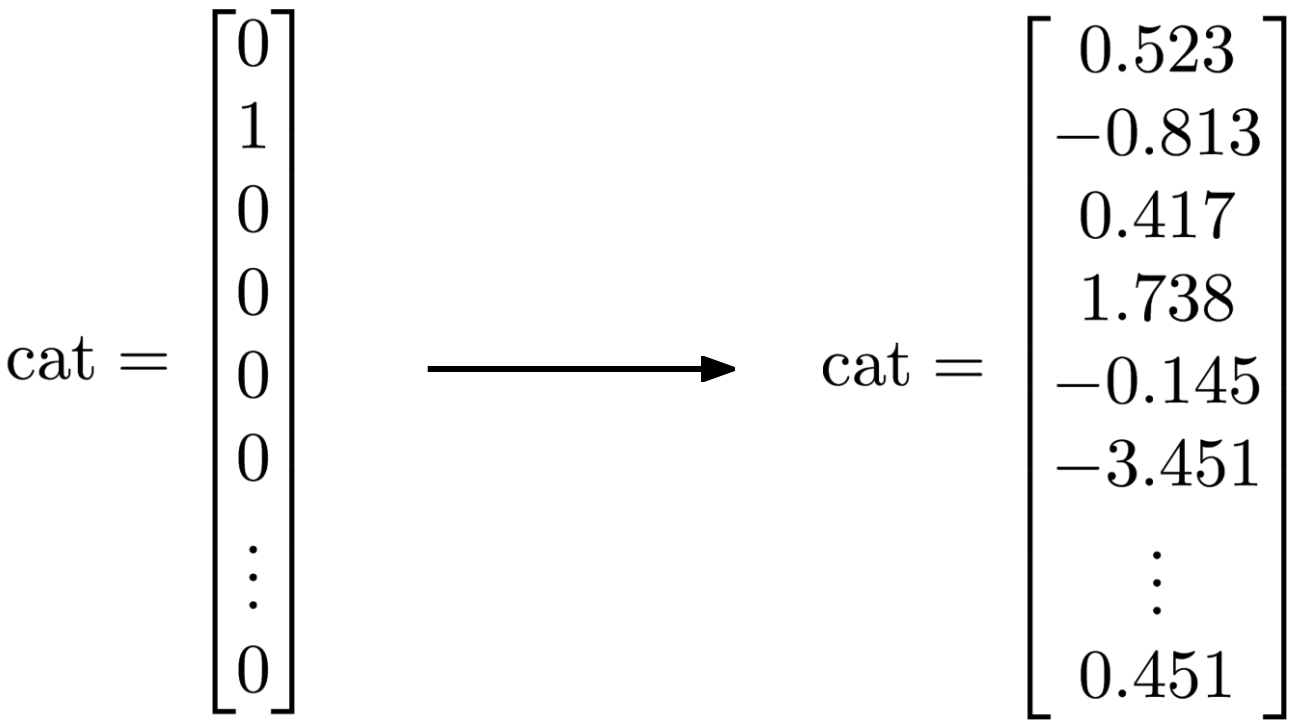
Word Embeddings

- In one-hot vector representation, a word is represented as one large ***sparse*** vector
- Instead, **word embeddings** are ***dense*** vectors in some vector space

word vectors are ***continuous*** representations of words

vectors of different words give us information about the potential relations between the words - words closer together in meaning have vectors closer to each other

Word Embeddings



Word Embeddings

"Representation of words in continuous space"

Inherit benefits

- Reduce dimensionality
- Semantic relatedness
- Increase expressiveness
  - one word is represented in the form of several features (numbers)

Word Embeddings

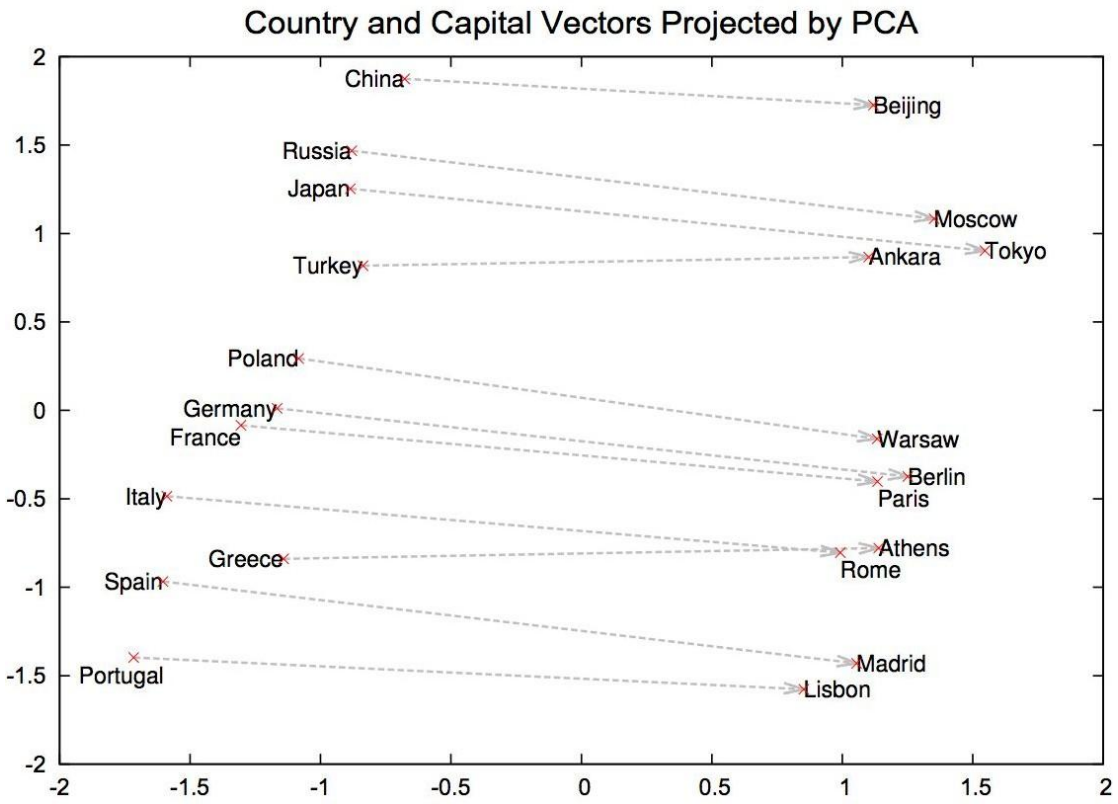
Play with some embeddings!

https://rare-technologies.com/word2vec-tutorial/#bonus_app

Try various relationships...

Word Embeddings

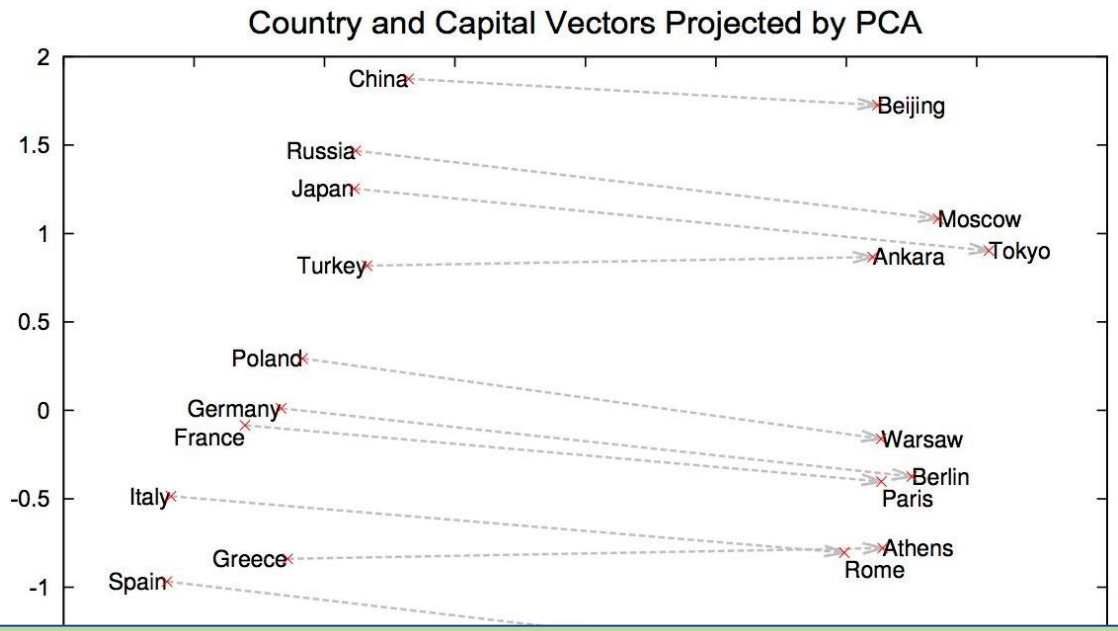
- Plot the embedding vectors



Mikolov et al. 2013

Word Embeddings

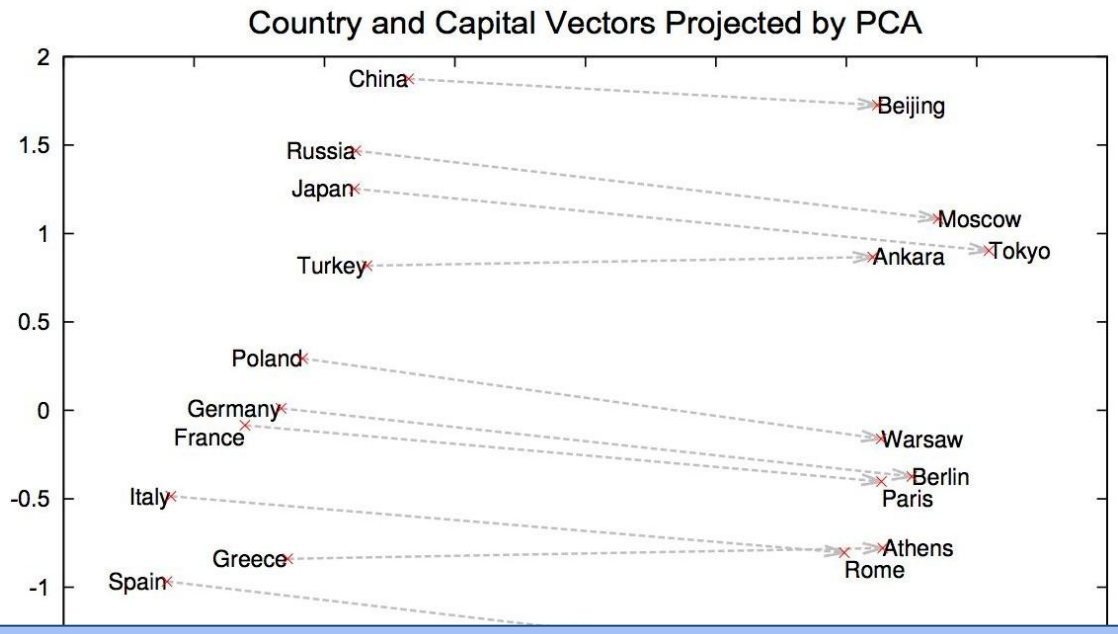
- Plot the embedding vectors



Plot shows the relationship between vectors representing related concepts

Word Embeddings

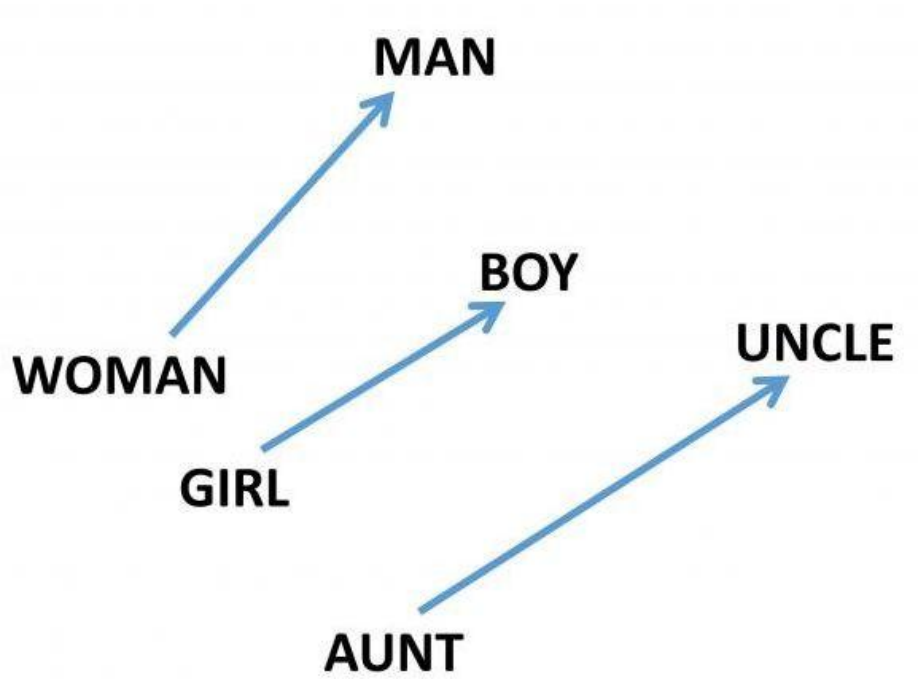
- Plot the embedding vectors



The vectors from countries to capitals point roughly in the same direction

Word Embeddings

- Similarly, learning the gender relationship



Word Embeddings

Q: How can we learn these embeddings automatically?

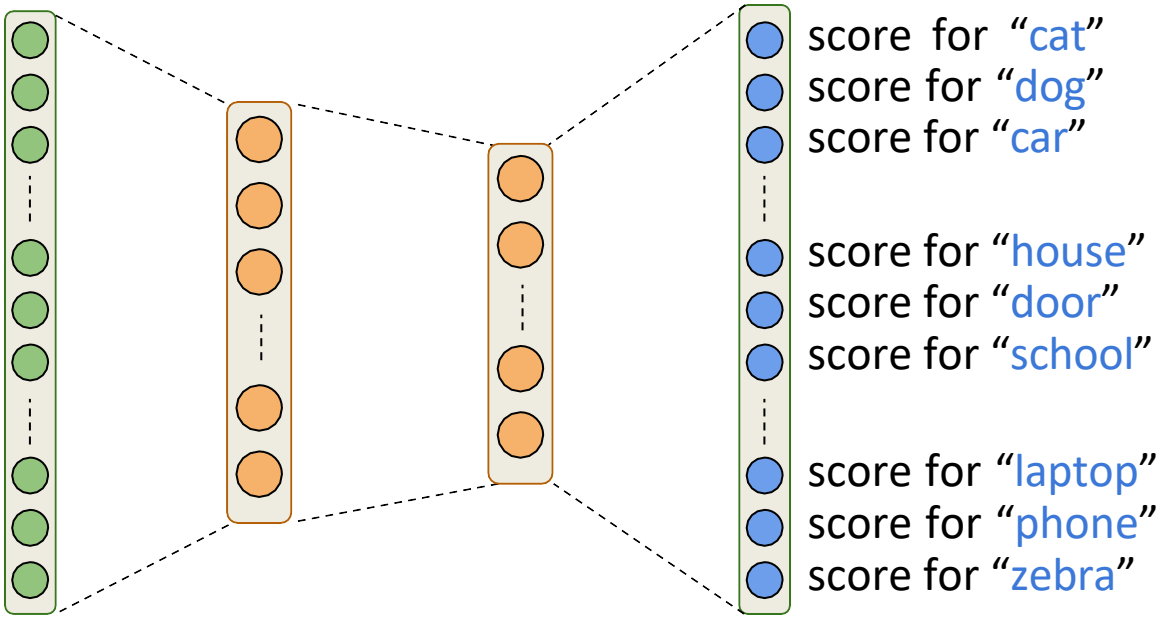
Word Embeddings

Q: How can we learn these embeddings automatically?

A: Neural Networks are a step ahead - embeddings are already learned as “richer” features

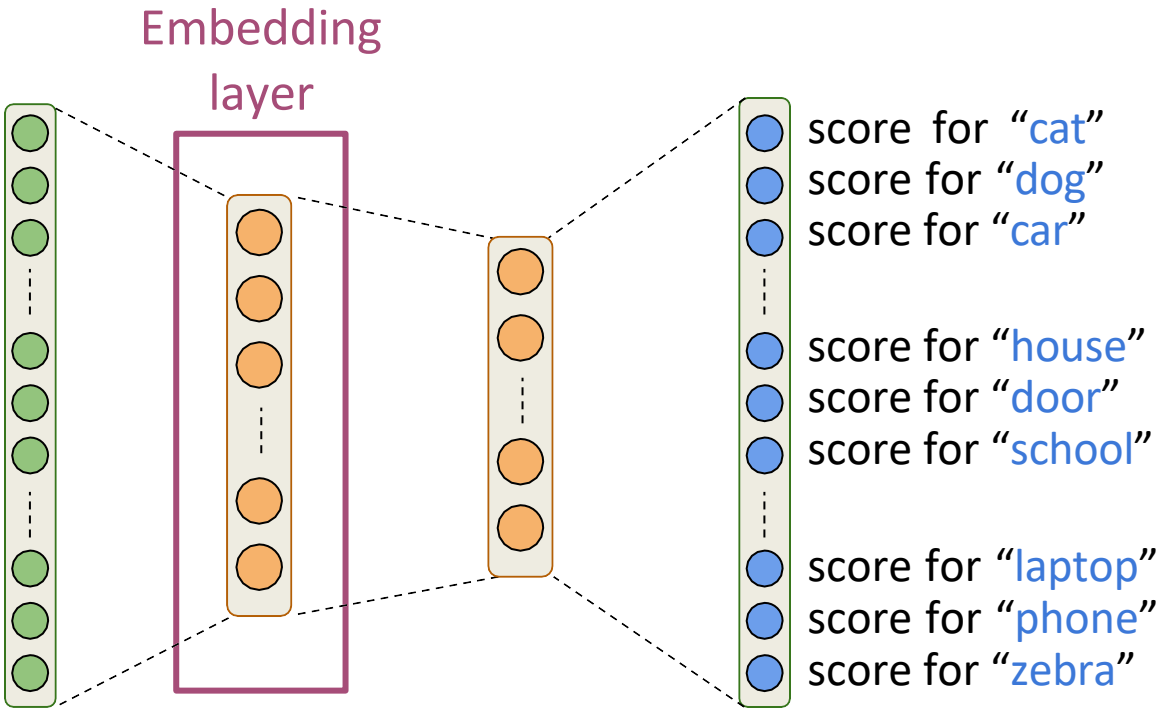
Word Embeddings

Neural Networks are a step ahead - embeddings are already learned as “richer” features



Word Embeddings

Neural Networks are a step ahead - embeddings are already learned as “richer” features



Word Embeddings

The overall **training task** defines the relationships which will be learned by the model

For example:

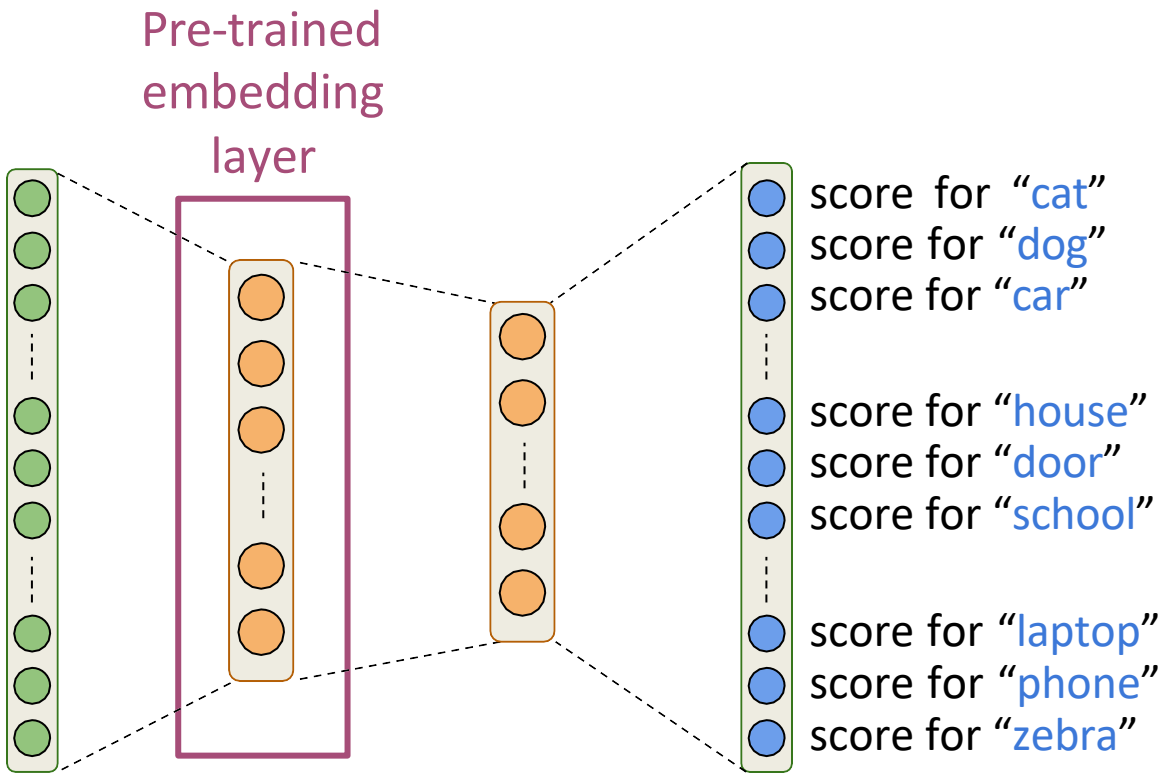
- In language modeling, the model uses neighboring context thus bringing words with similar context closer
- In doing POS tagging task, words with similar POS tags will come close to each other
- If our network is doing machine translation, the embeddings will be tuned for translation

Word Embeddings

- Generally, task specific embeddings are better than generic embeddings
- In case of small amount of training data, generic embeddings learned on large amount of data works better
- Generic embeddings can also be used as a starting point

Word Embeddings

We can use pre-trained embeddings as well - just initialize the weights in the first layer with some learned embeddings



Word Embedding Tools

Some tools to learn word embeddings:

- Word2Vec (from Google)
- FastText (from Facebook)
- GloVe (from Stanford)

Word Embeddings

A few pre-trained word embeddings

- GloVe: Wikipedia plus Gigaword https://goo.gl/1XYZhc
- FastText: Wikipedia of 294 languages https://goo.gl/1v423g
- Dependency-based https://goo.gl/tpgw4R

Using pre-trained embeddings in keras:

https://blog.keras.io/using-pre-trained-word-embeddings-in-a-keras-model.html

Summary

- Neural networks
  - Activation function
  - Forward pass, loss, backward pass, update
  - Implementation in Keras
- Neural network language model
  - Input representation
  - Output representation
- Word embeddings
- Neural network language model in Keras

Neural Network Implementation

Let's implement!

- Neural network
  - Spiral data
- Neural network language model
  - Sherlock holmes data
- Neural network for multiclass classification
  - Sentiment analysis (14 way classification)
    - one-hot vector for every sentence