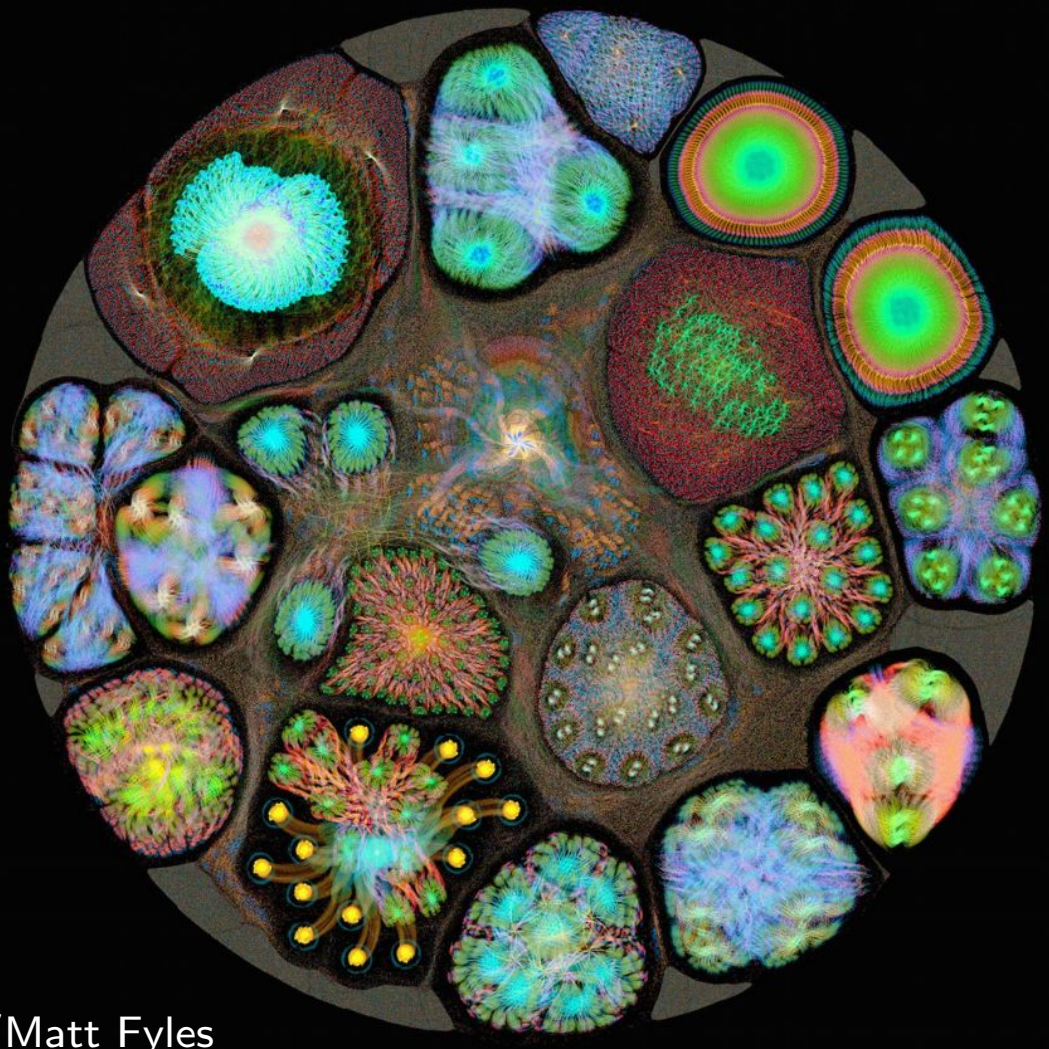
An Intro to Deep Learning

Emmanuel Bengio

June 6, 2017

What is deep learning?



Credit: Graphcore/Matt Fyles

What is deep learning?

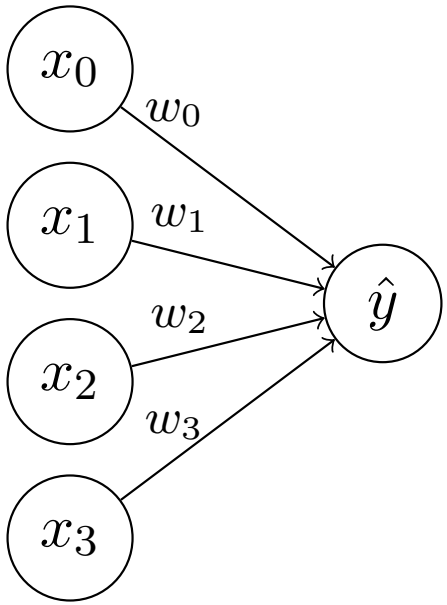
The technical answer:

- Stacking layers of linear transforms and non-linearities

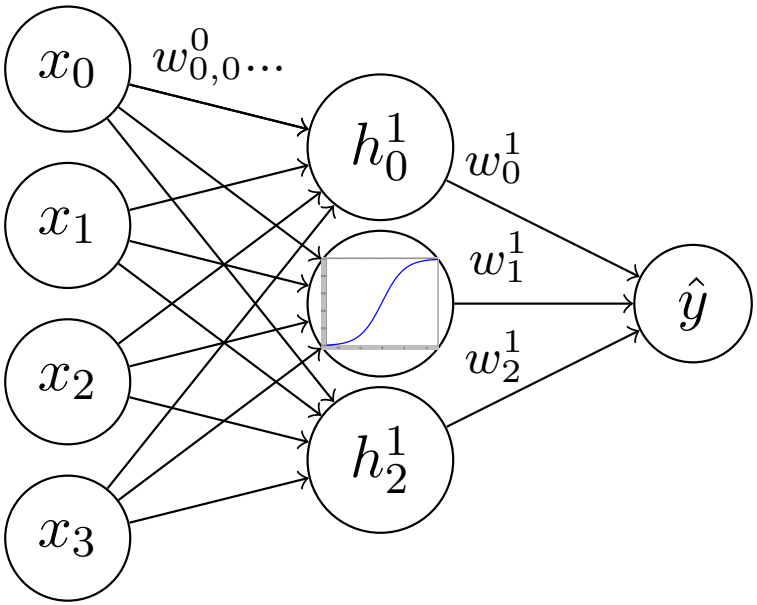
The less-technical answer:

- Stacking layers of abstractions
- Brain-like structure
- Learning **representations**

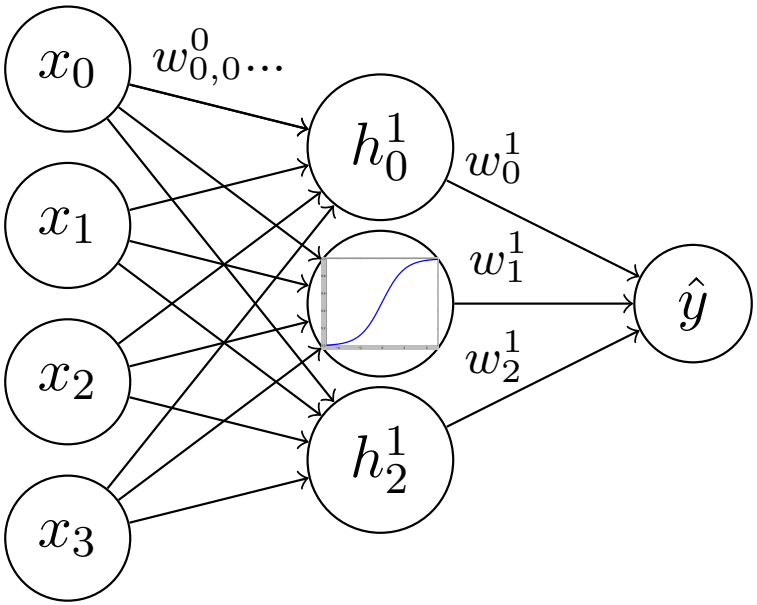
Linear regression



Adding intermediate nodes

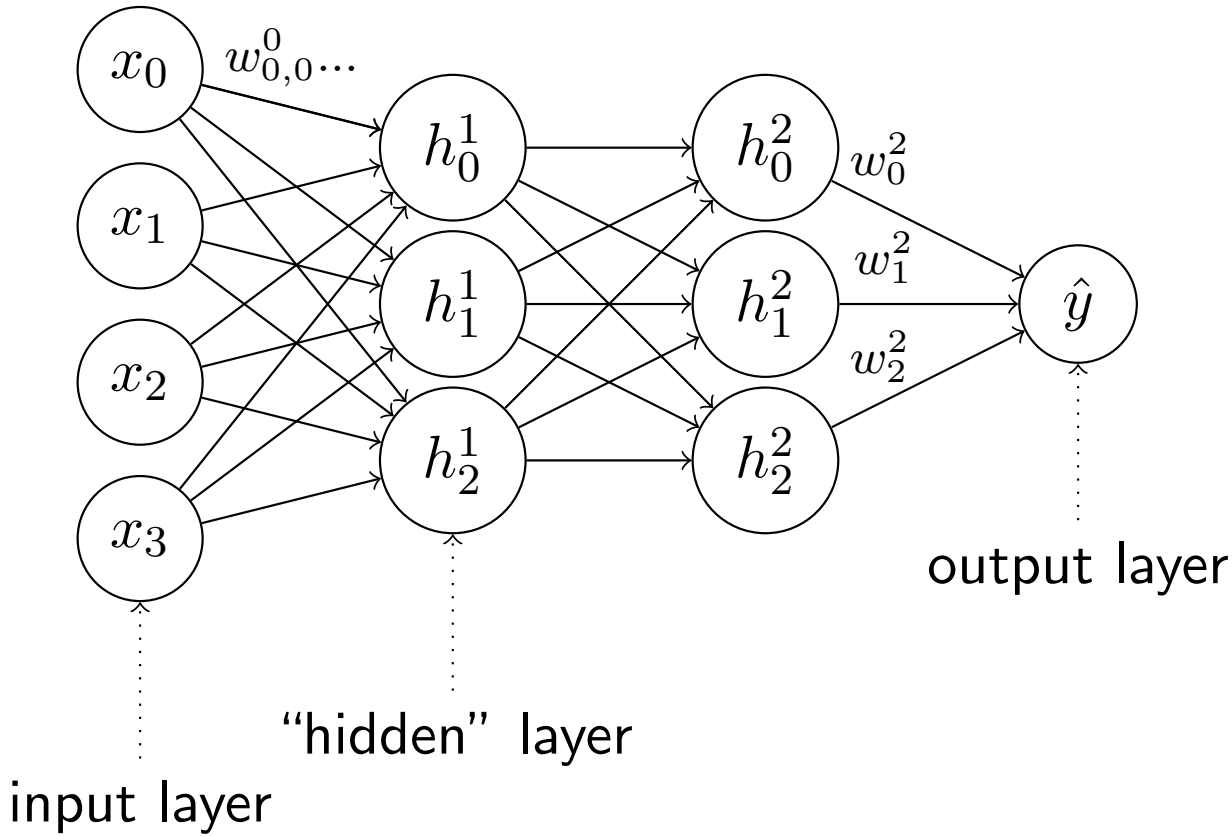


Adding intermediate nodes



(By the way with this you can theoretically represent any function!)

Adding more layers



In math

In terms of linear algebra, no hidden layers (linear regression):

$$\hat{y} = \mathbf{w}^\top \mathbf{x}$$

In math

In terms of linear algebra, no hidden layers (linear regression):

$$\hat{y} = \mathbf{w}^\top \mathbf{x}$$

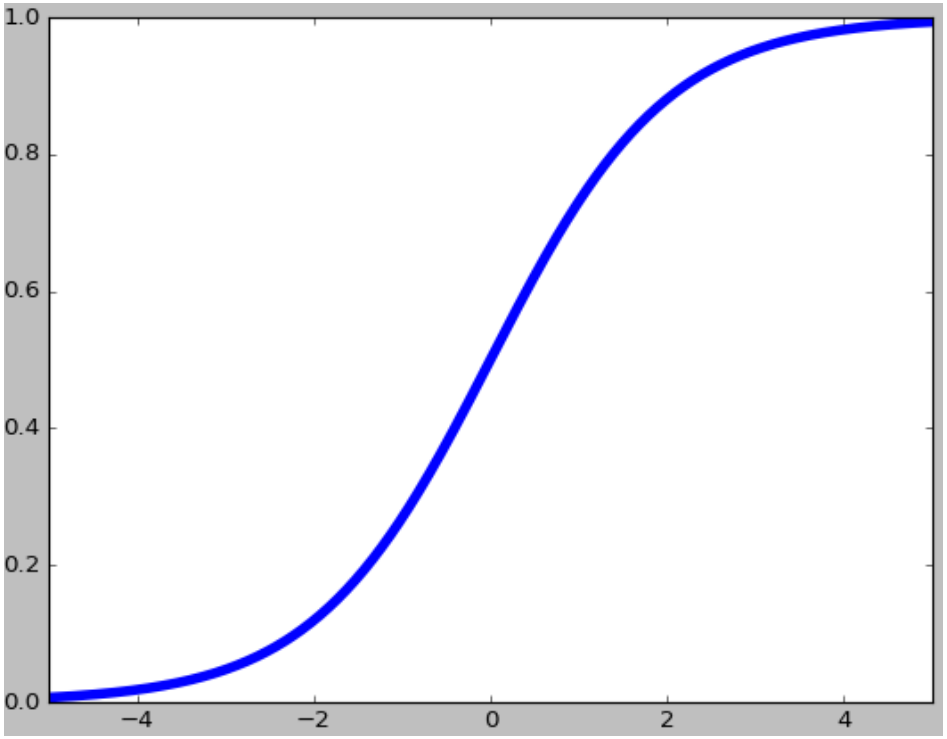
A single hidden layer:

$$\hat{y} = \mathbf{w}^\top f(W\mathbf{x} + \mathbf{b})$$

Activation Functions

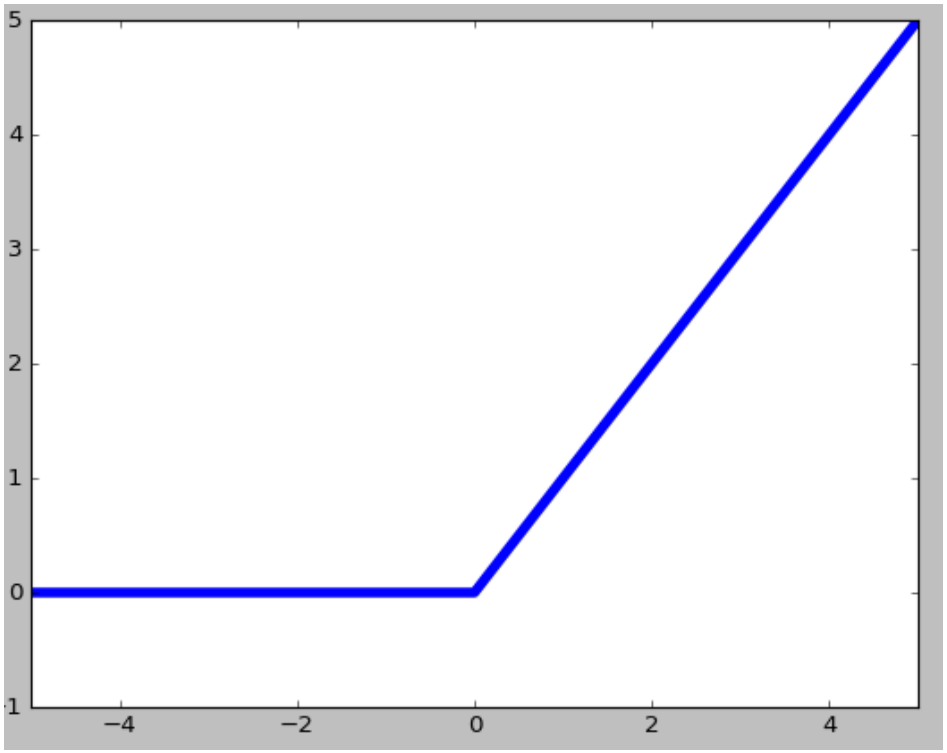
Activation functions are applied *element-wise* to each element of a vector. For example the sigmoid:

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



Right now the most popular activation function is the Rectified Linear Unit (ReLU):

$$f(x) = \max\{0, x\}$$



Training Neural Networks via Gradient Descent

Training NNets is conceptually rather simple:

- Show an example x
- Compute $\hat{y}$ and the error (e.g. $E = (\hat{y} - y)^2$)
- Adjust the weights w by $\alpha \frac{\partial E}{\partial w}$
- Repeat

α is called a learning rate.

Training Neural Networks via Gradient Descent

$$\frac{\partial E}{\partial w} \text{ or } \nabla_w E$$

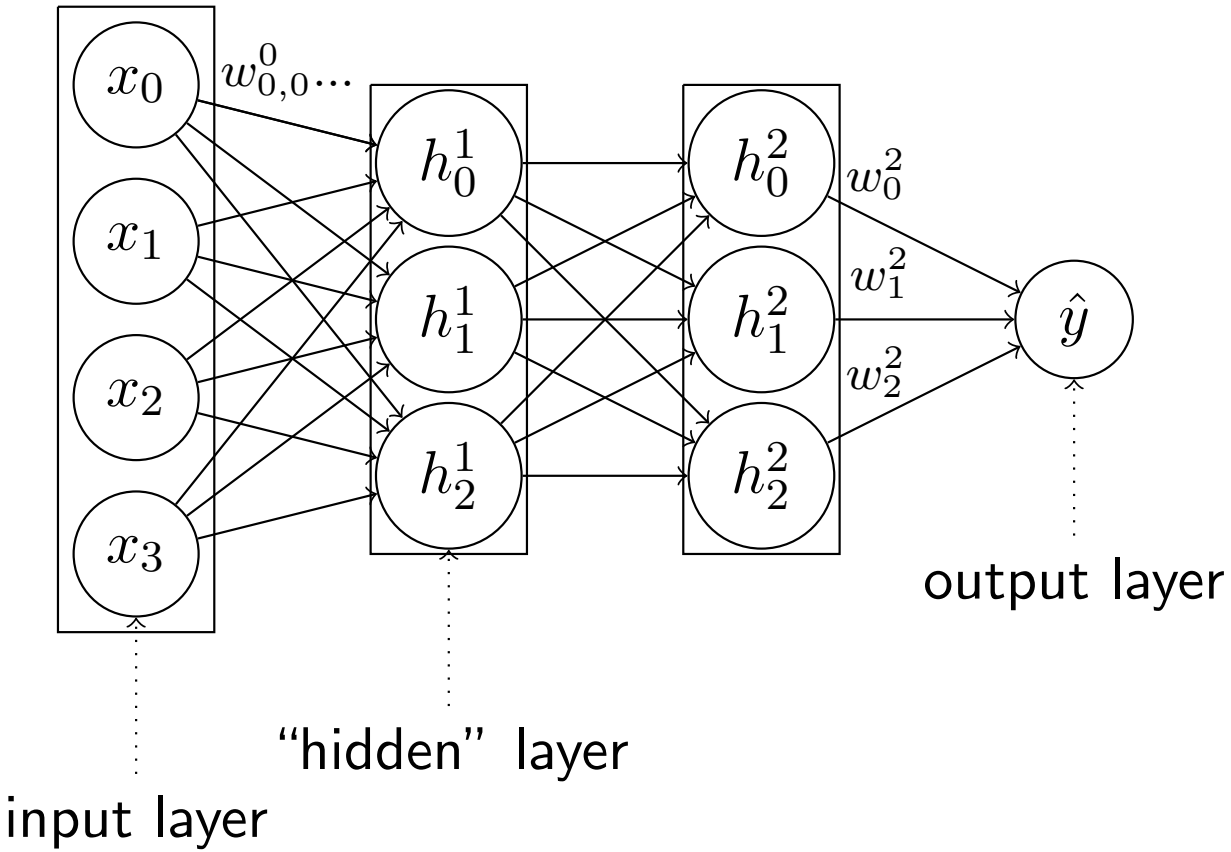
is called the gradient of the error with respect to the parameters.
Deep learning libraries compute it for you!

Deep Learning Software

- Theano, Tensorflow, MXNet, PyTorch
- Keras, Lasagne

Theano is older but stable. Tensorflow is more recent, engineered-feeling. MXNet/PyTorch are less verbose and also more recent.
Keras and Lasagne are built on top of these libraries, abstract NNet operations.

A sequence of blocks



A sequence of blocks

```
import keras
from keras.models import Sequential
from keras.layers import Dense
```

A sequence of blocks

```
model = Sequential()
model.add(Dense(32, activation='relu',
                    input_shape=10))
model.add(Dense(64, activation='relu'))
model.add(Dense(10, activation='softmax'))
```

Compiling the model

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

Training the model via gradient descent

```
model.fit(X_train, Y_train,
          batch_size=32, epochs=10, verbose=1)
```

Evaluating the model on new data

```
score = model.evaluate(X_test, Y_test, verbose=0)
```