DEEP TRADING

THE PYTHON QUANTS

Yam Peleg

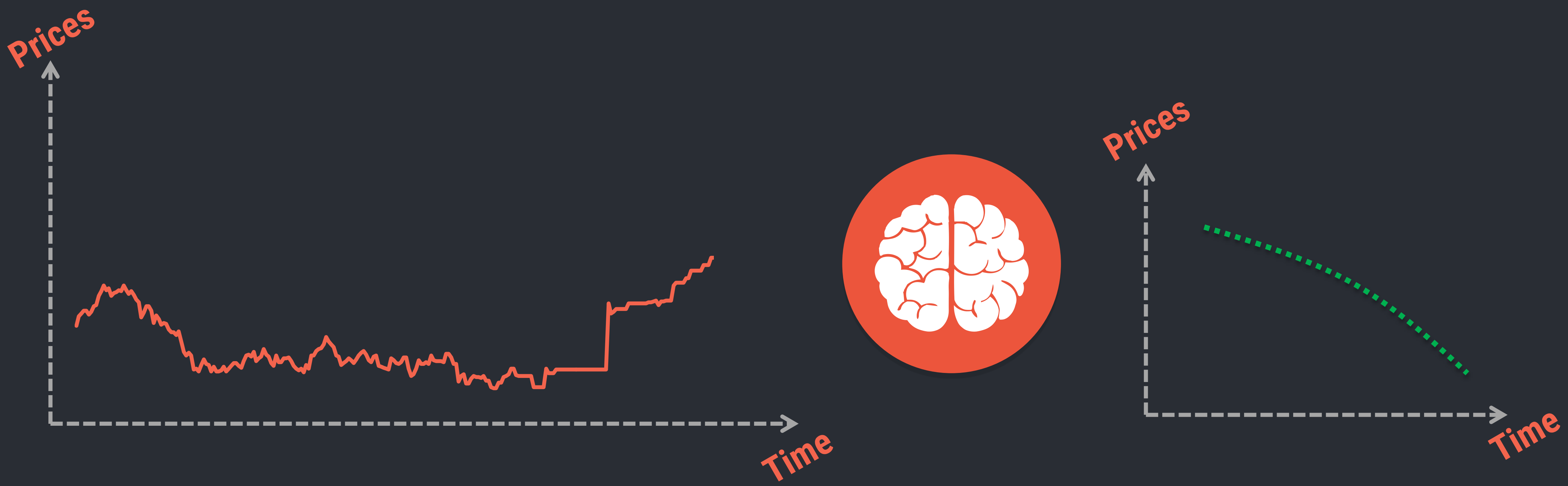
DEEP LEARNING FOR TRADING

OUR GOAL

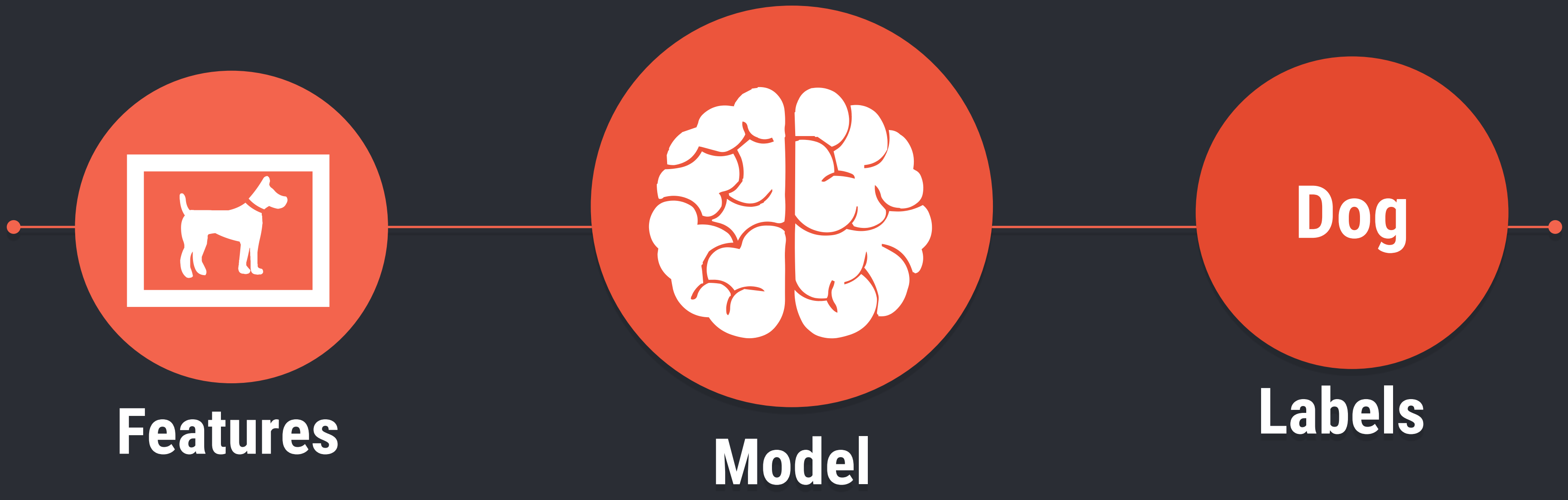
We want to predict the future.

OUR GOAL

We want to predict the future.



SUPERVISED LEARNING



Each example in the training data is a *pair* consisting of an input vector (features) and a desired output value (labels).

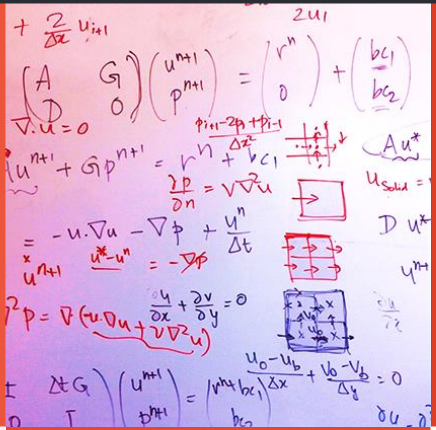
A supervised learning algorithm analyzes the training data and approximate a function, which can be used for mapping new unlabeled examples.

FINNANCIAL PREDICTION PITFALLS

The longer the time frame, the more difficult it will be to accurately forecast financial results.

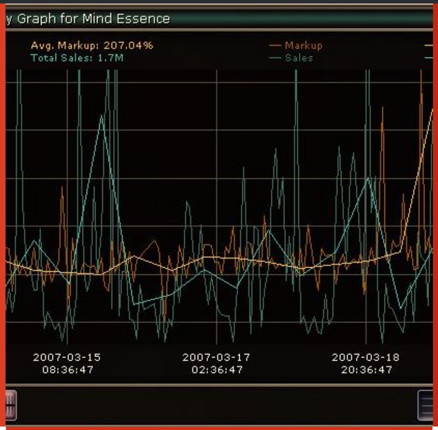
Importance

Data Importance is questionable and determination of meaningful data is hard.



Overfitting

Overfitted easily, most models have poor predictive capabilities On financial data.



Behavior

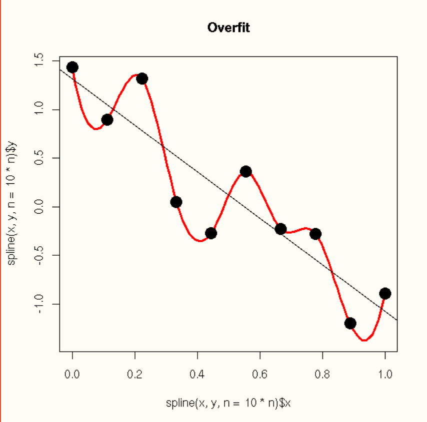
Behavior of financial markets change all the time and can be really unpredictable.

Much Data

Possible relevant data from many markets is incredibly large.

No Theory

Complex non–linear interactions in the data are not well specified by financial theory.



Noisy Data

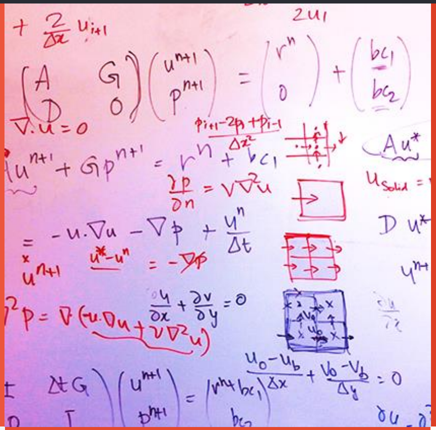
Noise In financial data Is very common and sometimes distinguishing noise from behavior is hard.

WHY DEEP LEARNING?

This is why.

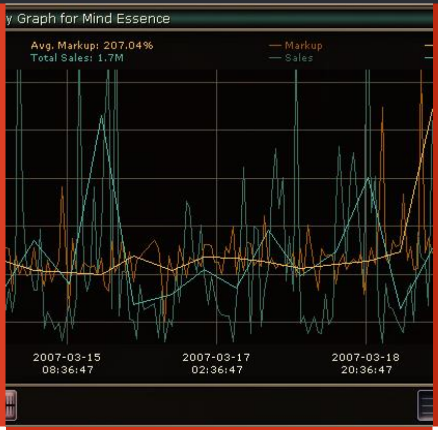
Importance

Data Importance is questionable and determination of meaningful data is hard.



Overfitting

Overfitted easily, most models have poor predictive capabilities On financial data.



Behavior

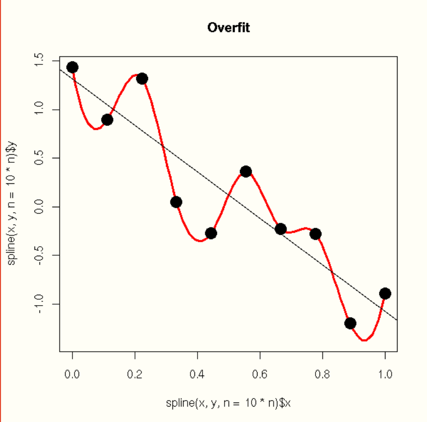
Behavior of financial markets change all the time and can be really unpredictable.

Much Data

Possible relevant data from many markets is incredibly large.

No Theory

Complex non-linear interactions in the data are not well specified by financial theory.

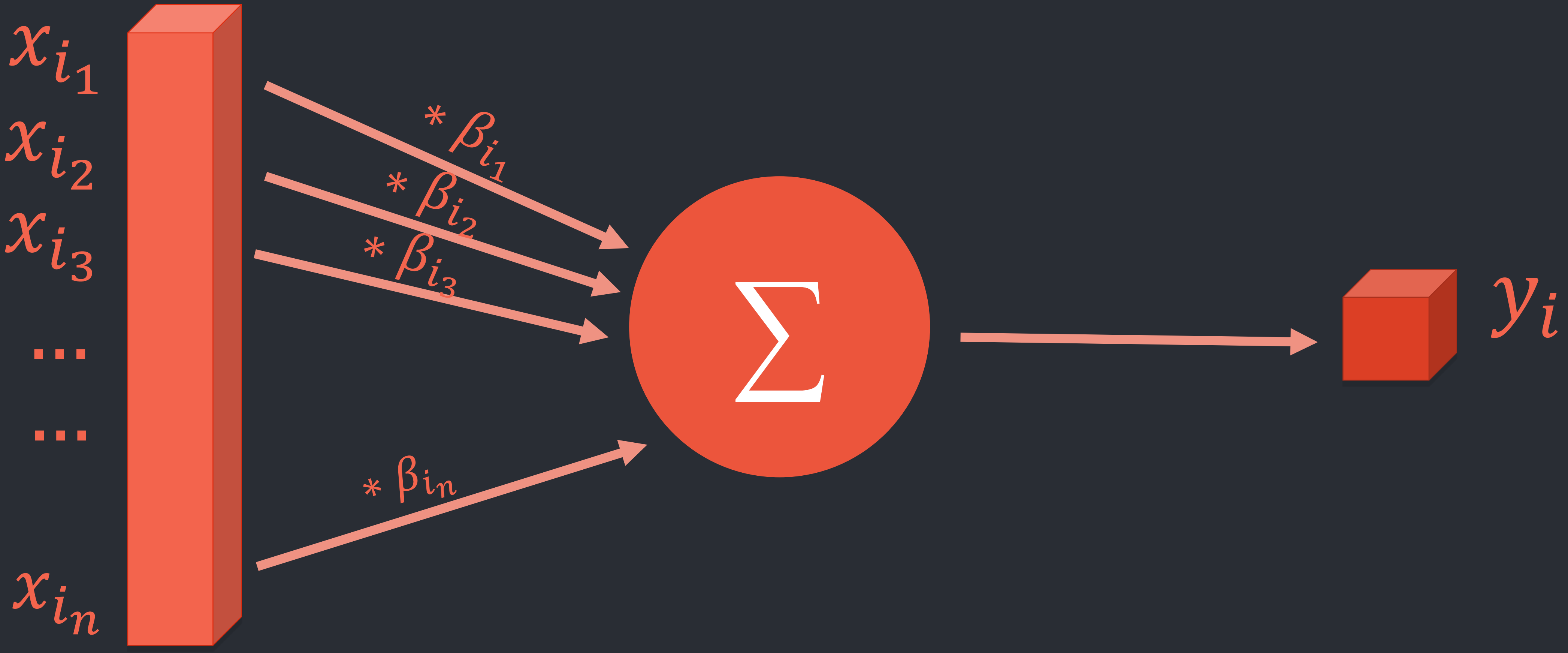


Noisy Data

Noise In financial data Is very common and sometimes distinguishing noise from behavior is hard.

LINEAR REGRESSION

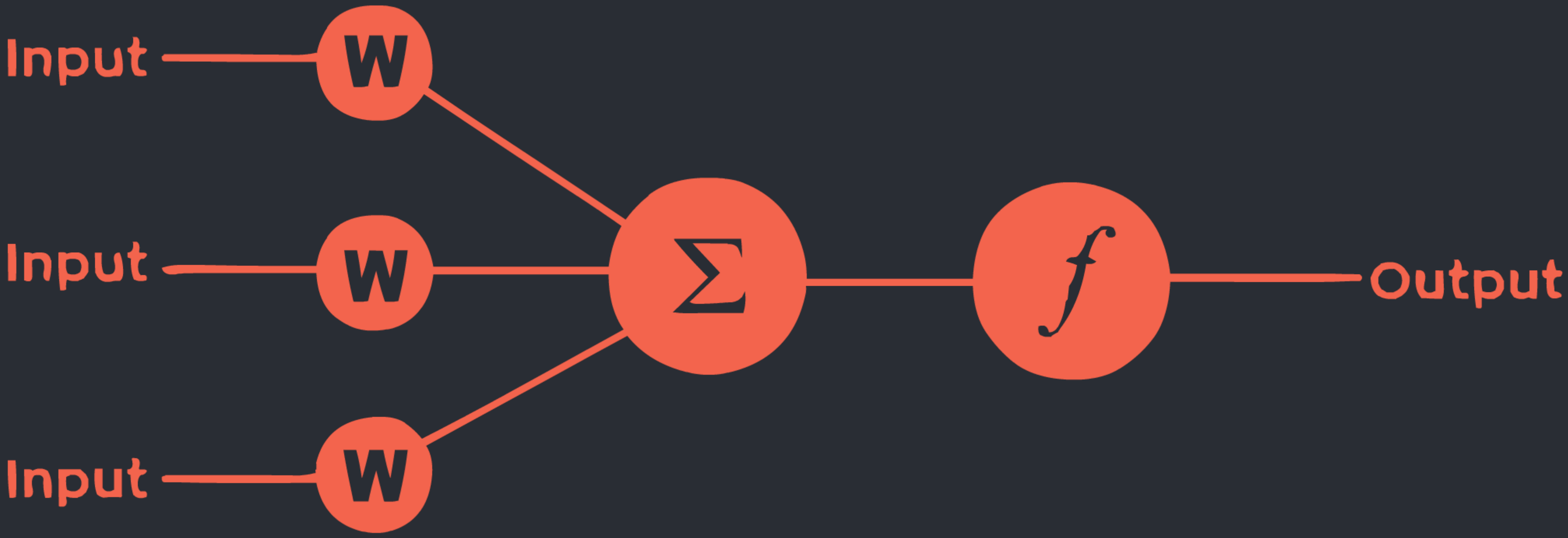
Regression



$$y_i = \beta_0 + \beta_1 x_{i_1} + \beta_2 x_{i_2} + \cdots + \beta_n x_{i_n} + \varepsilon$$

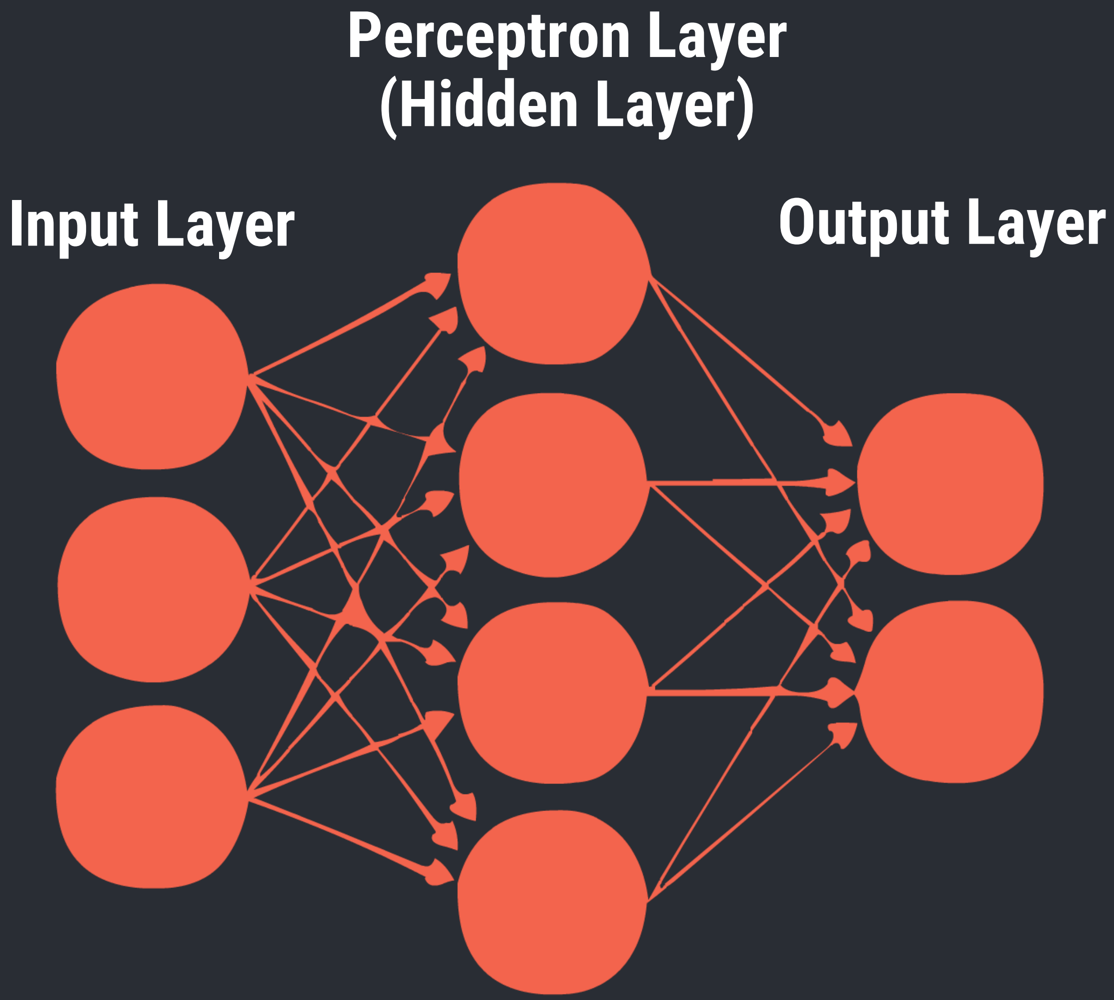
Perceptron

The Artificial Neuron



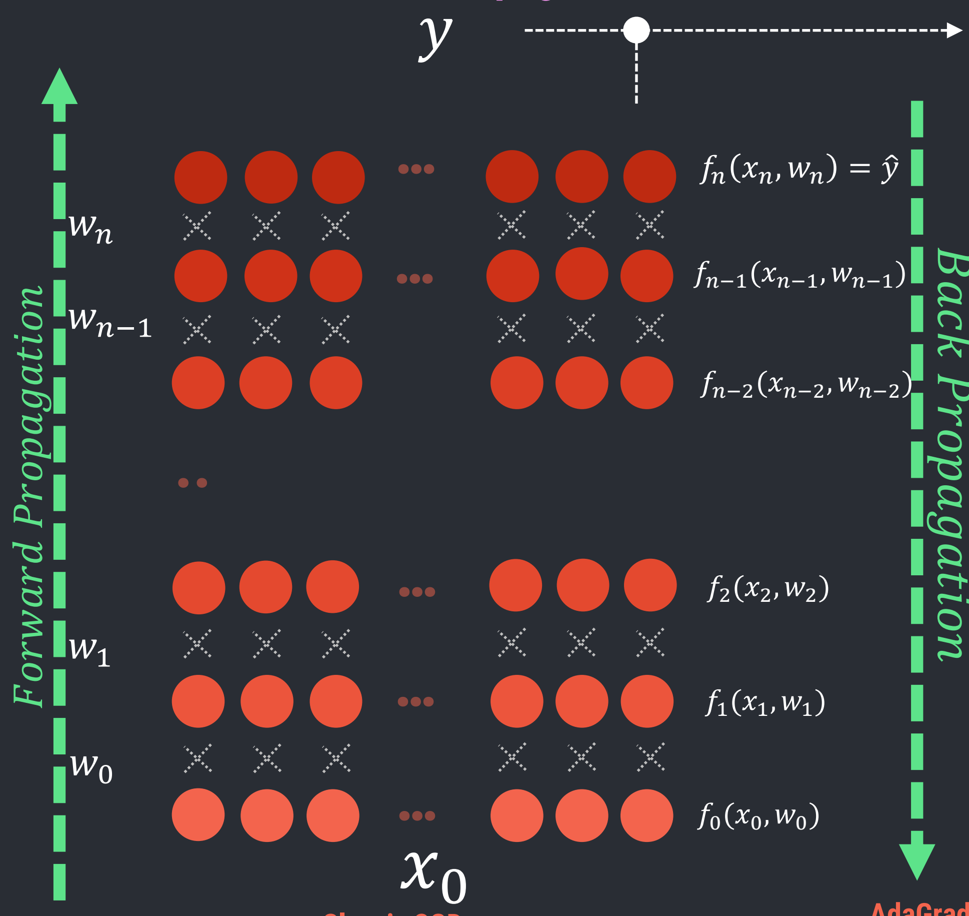
Neural Network

One Layer Perceptron



GRADIENT BASED MODELS

1: Forward Propagation



2: Loss Calculation

$$E = l(\hat{y}, y)$$

$$\frac{\partial E}{\partial x_n} = \frac{\partial l(\hat{y}, y)}{\partial x_n}$$

$$\frac{\partial E}{\partial x_{n-1}} = \frac{\partial E}{\partial x_n} \frac{\partial f_n(x_{n-1}, w_n)}{x_{n-1}}$$

$$\frac{\partial E}{\partial w_n} = \frac{\partial E}{\partial x_n} \frac{\partial f_n(x_{n-1}, w_n)}{\partial w_n}$$

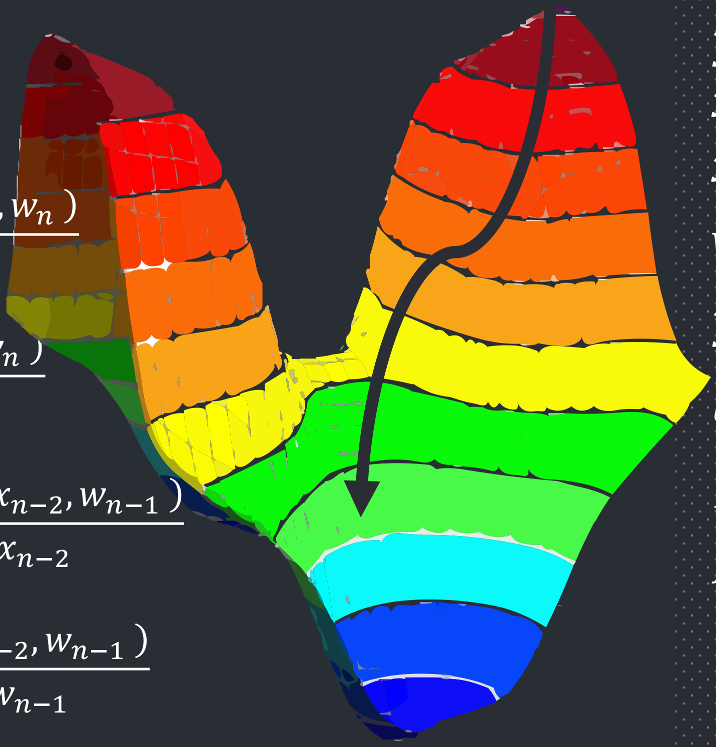
$$\frac{\partial E}{\partial x_{n-2}} = \frac{\partial E}{\partial x_{n-1}} \frac{\partial f_{n-1}(x_{n-2}, w_{n-1})}{x_{n-2}}$$

$$\frac{\partial E}{\partial w_{n-1}} = \frac{\partial E}{\partial x_{n-1}} \frac{\partial f_n(x_{n-2}, w_{n-1})}{\partial w_{n-1}}$$

...

...

3: Optimization



Legend

- y – Ground Truth
- x_0 - Features Vector
- x_i - Output of i layer
- w_i - Weights of i layer
- $\hat{y}$ – Model Output
- $l(\hat{y}, y)$ - Loss Function
- E – Loss Surface
- f – Activation Function

Classic SGD

$$v_t = \mu v_{t-1} - \alpha \nabla L_t(w_{t-1})$$
$$w_t = w_{t-1} - V_t$$

AdaGrad

$$w_t = w_{t-1} - \alpha \frac{\nabla L_t(w_{t-1})}{\sqrt{\sum_{t'=1}^{t} \nabla L_{t'}(w_{t'-1})^2}}$$

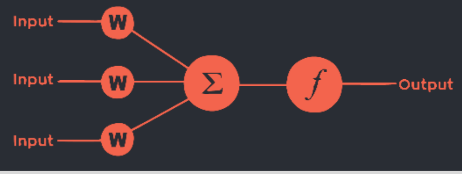
RMSProp

$$R_t = \gamma R_{t-1} + (1 - \gamma) \nabla L_t(w_{t-1})^2$$
$$w_t = w_{t-1} - \alpha \frac{\nabla L_t(w_{t-1})}{\sqrt{R_t}}$$

Adam

$$M_t = \frac{\beta_1 M_{t-1} + (1 - \beta_1) \nabla L_t(w_{t-1})}{(1 - \beta_1)^t}$$
$$R_t = \frac{\beta_2 M_{t-1} + (1 - \frac{\beta_2}{2}) \frac{\nabla L_t(w_{t-1})}{2}}{(1 - \beta_2)^2}$$
$$w_t = w_{t-1} - \alpha \frac{M_t}{\sqrt{R_t}}$$

DEEP LEARNING COMMON STRUCTURES



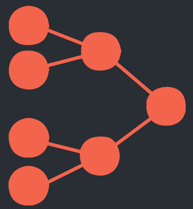
Perceptron It is a type of linear classifier, a classification algorithm that makes its predictions based on a linear predictor function combining a set of weights with the feature vector. The algorithm allows for online learning, in that it processes elements in the training set one at a time.

SUPERVISED

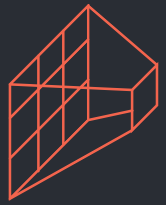
FEED FORWARD

Feed Forward Network sometimes Referred to as MLP, is a fully connected dense model used as a simple classifier.

Convolutional Network assume that highly correlated features located close to each other in the input matrix and can be pooled and treated as one in the next layer.



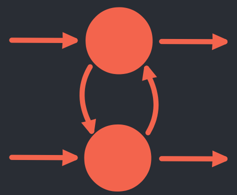
Known for superior Image classification capabilities.



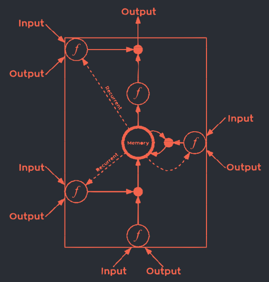
RECURRENT

Simple Recurrent Neural Network is a class of artificial neural network where connections between units form a directed cycle.

Hopfield Recurrent Neural Network It is a RNN in which all connections are symmetric. it requires stationary inputs.

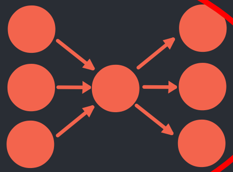


Long Short Term Memory Network contains gates that determine if the input is significant enough to remember, when it should continue to remember or forget the value, and when it should output

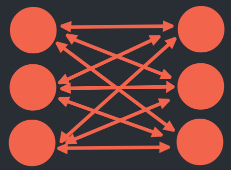


UNSUPERVISED

Auto Encoder aims to learn a representation (encoding) for a set of data, typically for the purpose of dimensionality reduction.



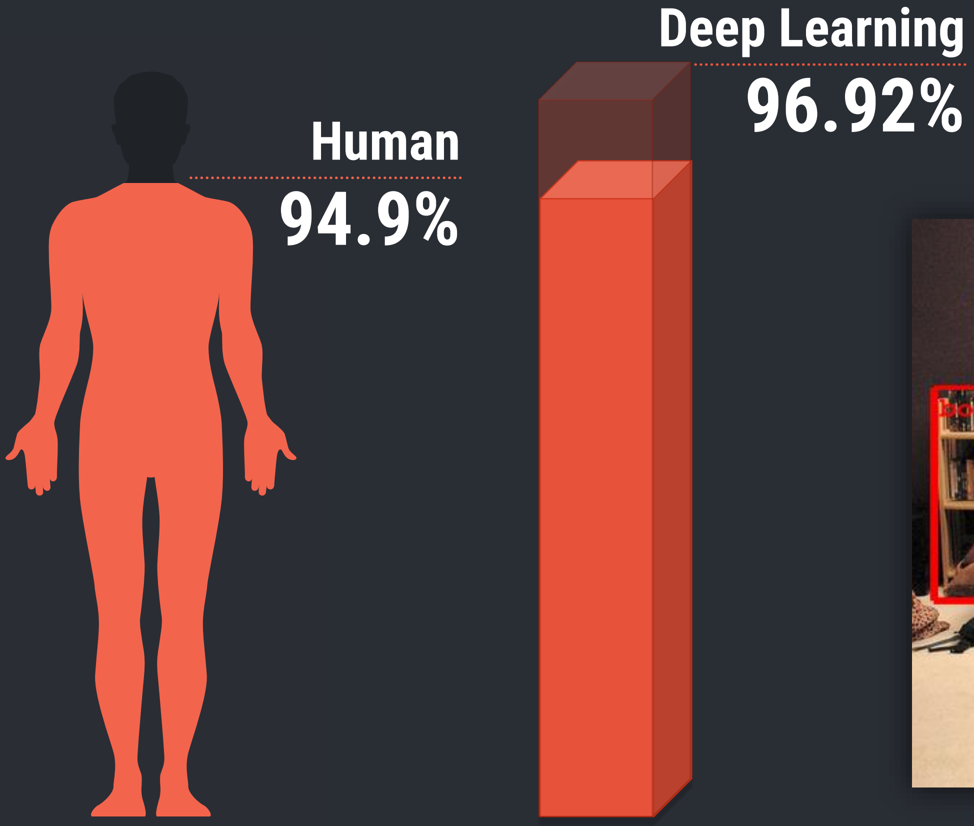
Restricted Boltzmann Machine can learn a probability distribution over its set of inputs..

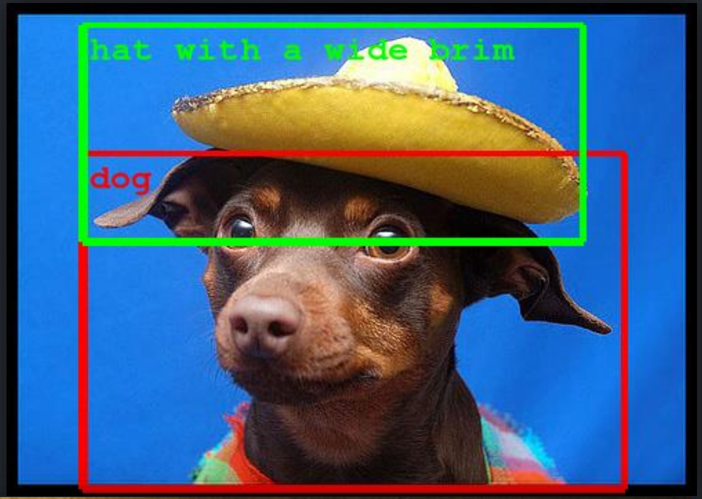


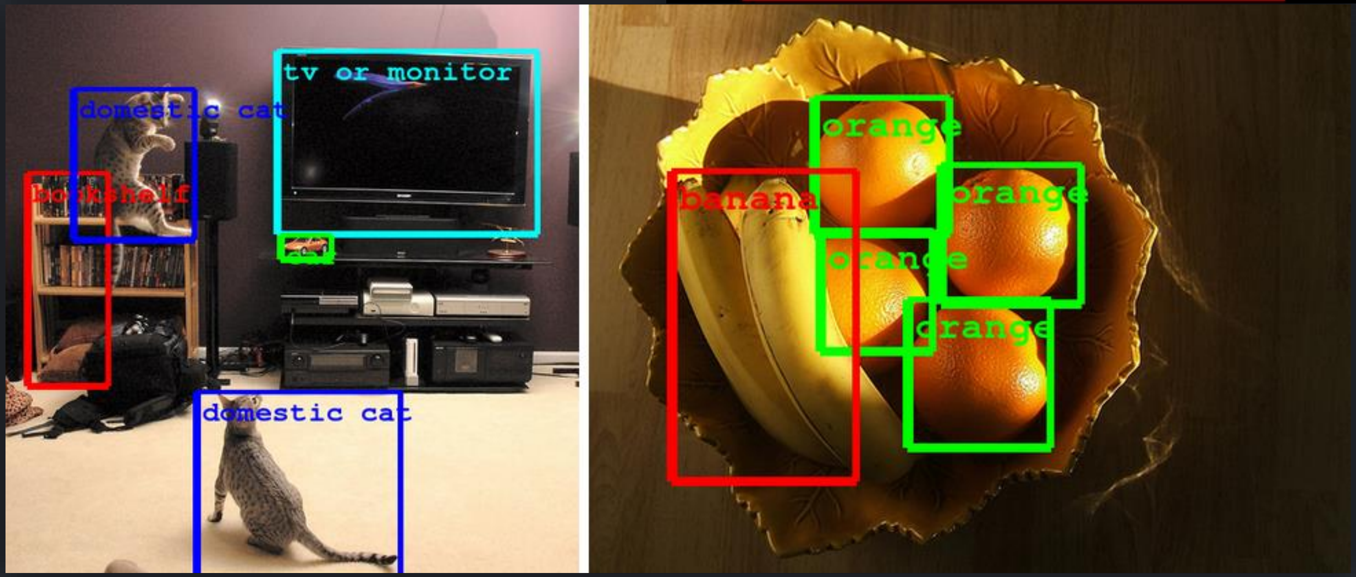
Deep Belief Net is a composition of simple, unsupervised networks such as restricted Boltzmann machines ,where each sub-network's hidden layer serves as the visible layer for the next.

DEEP LEARNING SUPERIORITY

Deep Learning is better then humans on certain Image recognition tasks.







ref: http://www.image-net.org/challenges/LSVRC/

Deep Neural Networks

For complex function approximation

Features

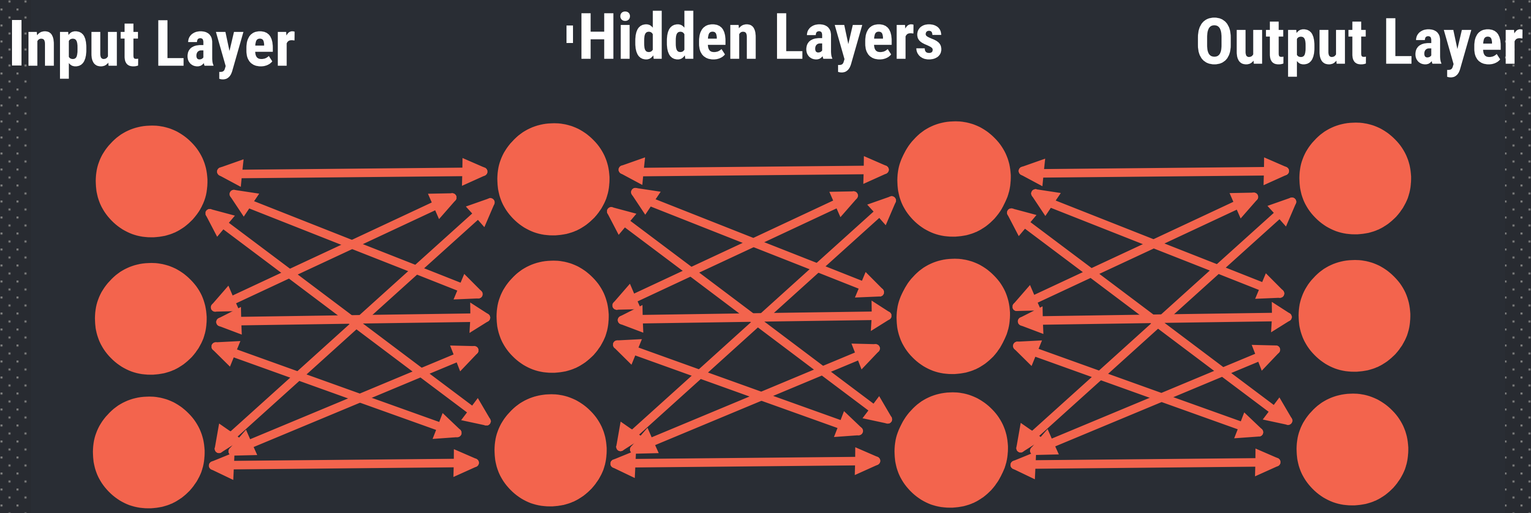
Past Prices

Correlations

Technical Analysis

Z Score

Time Features



Ground Truth

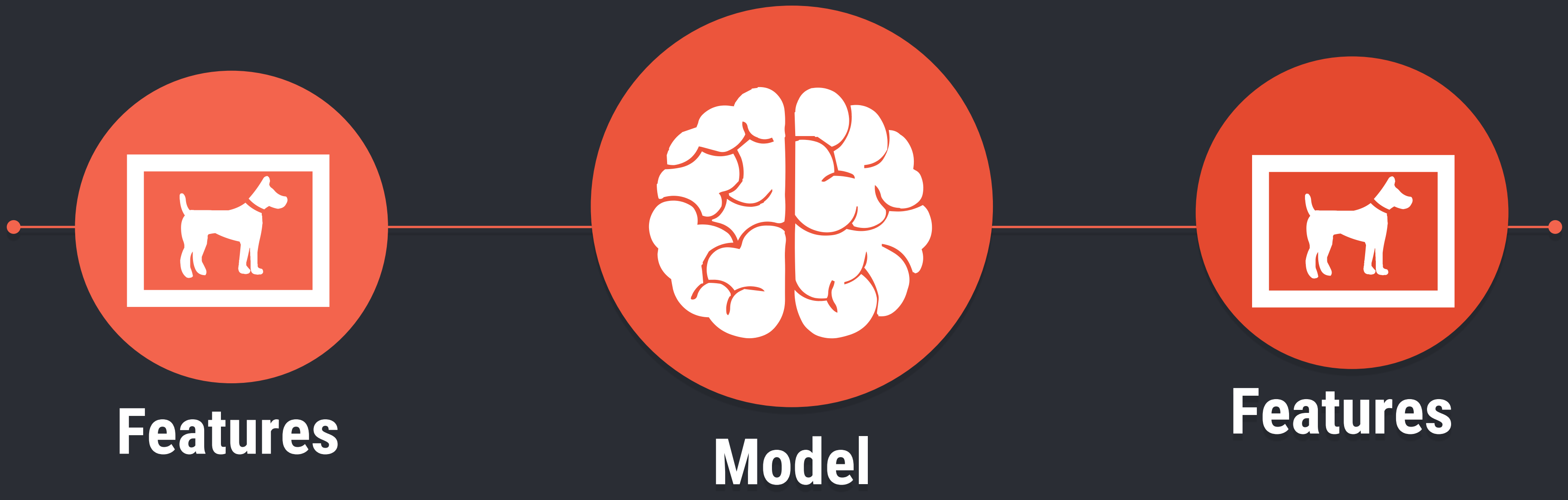
Future Prices
Regression

Up or Down
Classification

Recommended Papers

Implementing deep neural networks for financial market prediction, Dixon et al, 2015

UNSUPERVISED LEARNING



Each example in the training data is a *pair* consisting of an input vector and again the input vector.

The goal is to learn function that describes the hidden structure from unlabeled data.

Auto Encoders

For learning the distribution of the features

Features

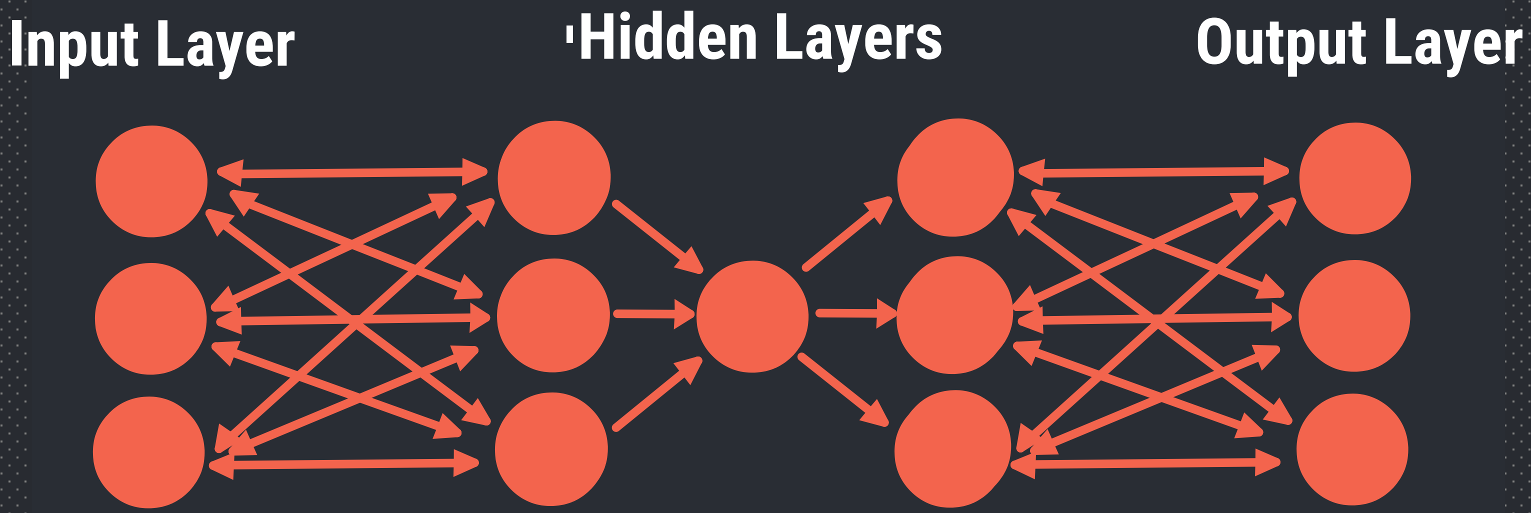
Past Prices

Correlations

Technical Analysis

Z Score

Time Features



Features

Past Prices

Correlations

Technical Analysis

Z Score

Time Features

Recommended Papers

Deep Modeling Complex Couplings within Financial Markets, Cao et al, AAAI 2015

Unsupervised Pretraining

For better approximation

Features

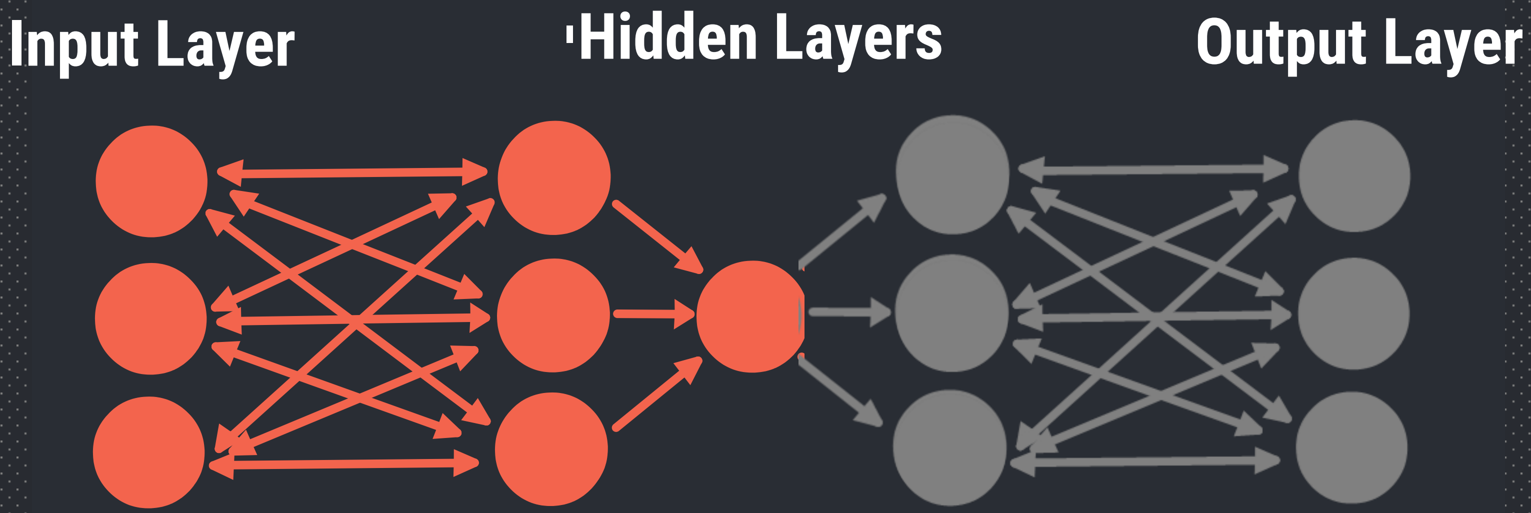
Past Prices

Correlations

Technical Analysis

Z Score

Time Features



Features

Ground Truth

Past Prices

Future Prices

Regression

Correlations

Technical Analysis

Up or Down

Classification

Z Score

Time Features

Recommended Papers

Applying Deep Learning to Enhance Momentum Trading Strategies in Stocks, L Takeuchi, 2013

Deep Learning for Multivariate Financial Time Series, Estrada, 2015

Yam Peleg

DEEP LEARNING WITH PYTHON

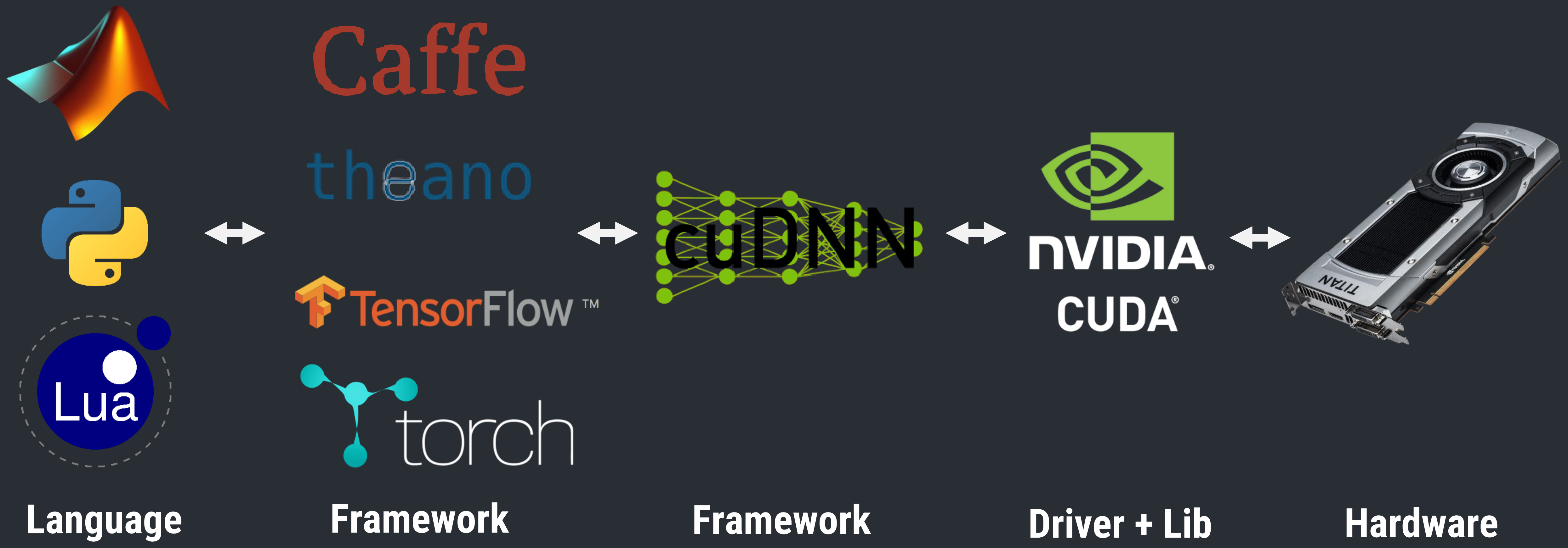
Deep Learning Hardware

Deep learning is often done on the GPU or other powerful devices

- Better for Matrix algebra
- Parallel calculations
- Much more powerful

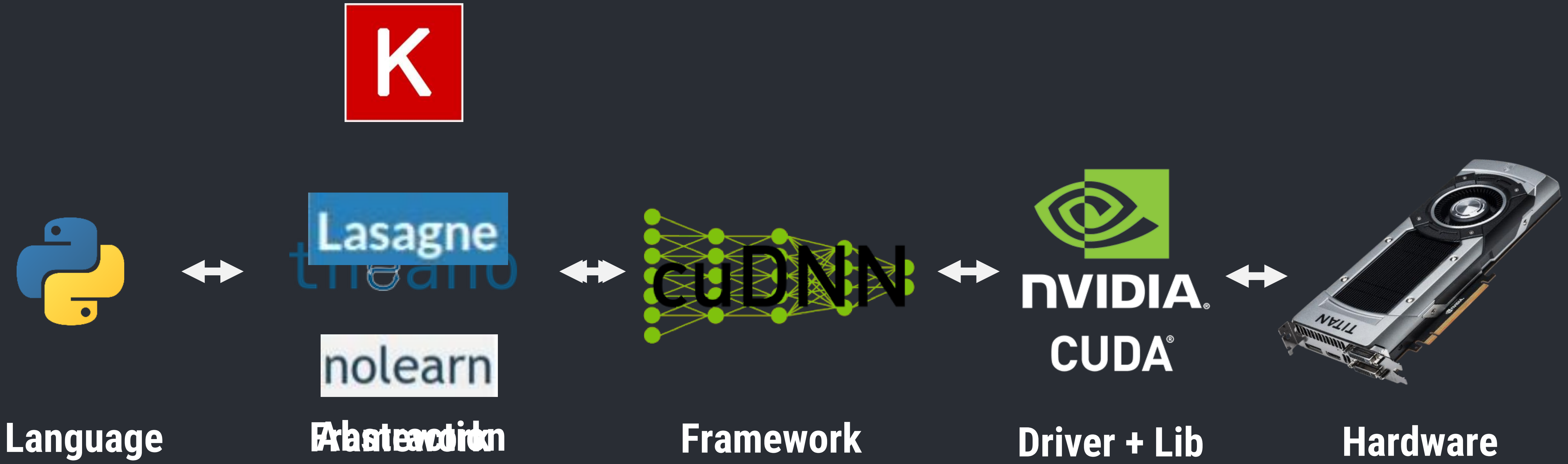
Deep Learning Framework

Deep learning is often done on the GPU or other powerful devices



Deep Learning Using Python

Deep learning is often done on the GPU or other powerful devices



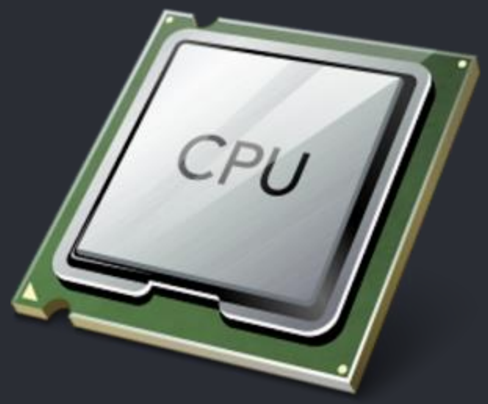
Python Stays Python

Deep learning is often done on the GPU or other powerful devices

```python
import theano.sandbox.cuda
theano.sandbox.cuda.use("gpu")
```

Theano

Theano is a Python library that allows you to define, optimize, and evaluate mathematical expressions involving multi-dimensional arrays efficiently.



Theano Tutorial

Language inside a Language

Theano Tutorial

Shared Variables

```
In [1]: import numpy, theano
        np_array = numpy.ones(2, dtype='float32')

        s_false   = theano.shared(np_array, borrow=False)
        s_true    = theano.shared(np_array, borrow=True)

        np_array += 1
        print(s_false.get_value())
        print(s_true.get_value())

Out [1]:[ 1.  1.]
        [ 2.  2.]
```

Variables

A Theano Variable is a Variable with storage that is shared between functions that it appears in.

Theano Tutorial

When using theano.function you're compiling C code performing your tasks under the hood. This is what makes **Theano fast.**

```
In [1]:  import theano
         x = theano.tensor.dscalar()
         f = theano.function([x], 2*x)
         f(4)

Out [1]: array(8.0)
```

Functions

The idea here is that we've compiled the symbolic graph (2*x) into a function that can be called on a number and will do some computations.

Theano Tutorial

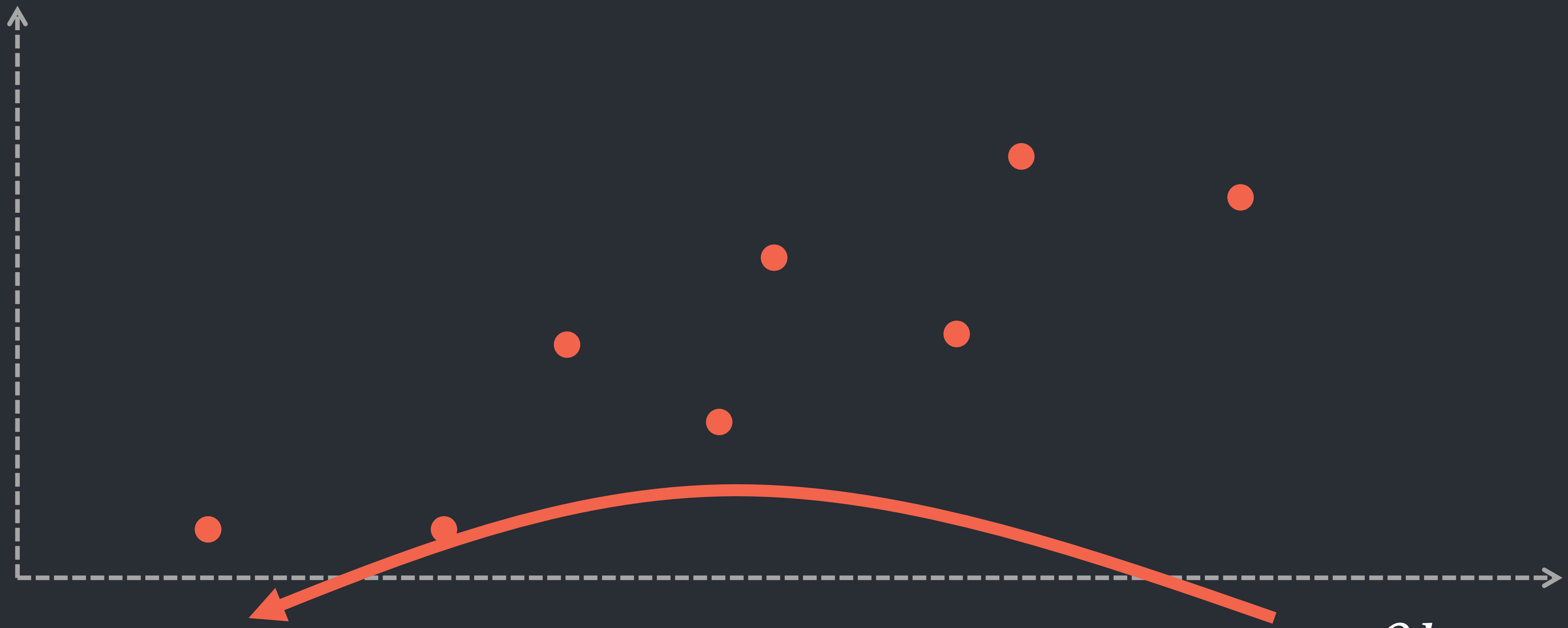
Gradients: computes the derivative of some expression

```
In [1]:  import numpy
         import theano
         import theano.tensor as T
         from theano import pp
         x = T.dscalar('x')
         y = x ** 2
         gy = T.grad(y, x)
         pp(gy)  # print out the gradient prior to optimization

Out [1]: '((fill((x ** TensorConstant{2}), TensorConstant{1.0}) *
         TensorConstant{2}) * (x ** (TensorConstant{2} - TensorConstant{1})))'

In [2]:  f = theano.function([x], gy)
         f(4)

Out [2]: array(8.0)
```

Gradients

Now let's use Theano for a slightly more sophisticated task: create a function which computes the derivative of some expression y with respect to its parameter x.

LINEAR REGRESSION



$$\hat{y} = wx \;\rightarrow\; l = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 \;\rightarrow\; w = w - \alpha\frac{\partial l}{\partial w}$$

Theano Tutorial

Gradient based linear regression

$$\hat{y} = wx$$

Function

$$l = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

Loss

$$w = w - \alpha\frac{\partial l}{\partial w}$$

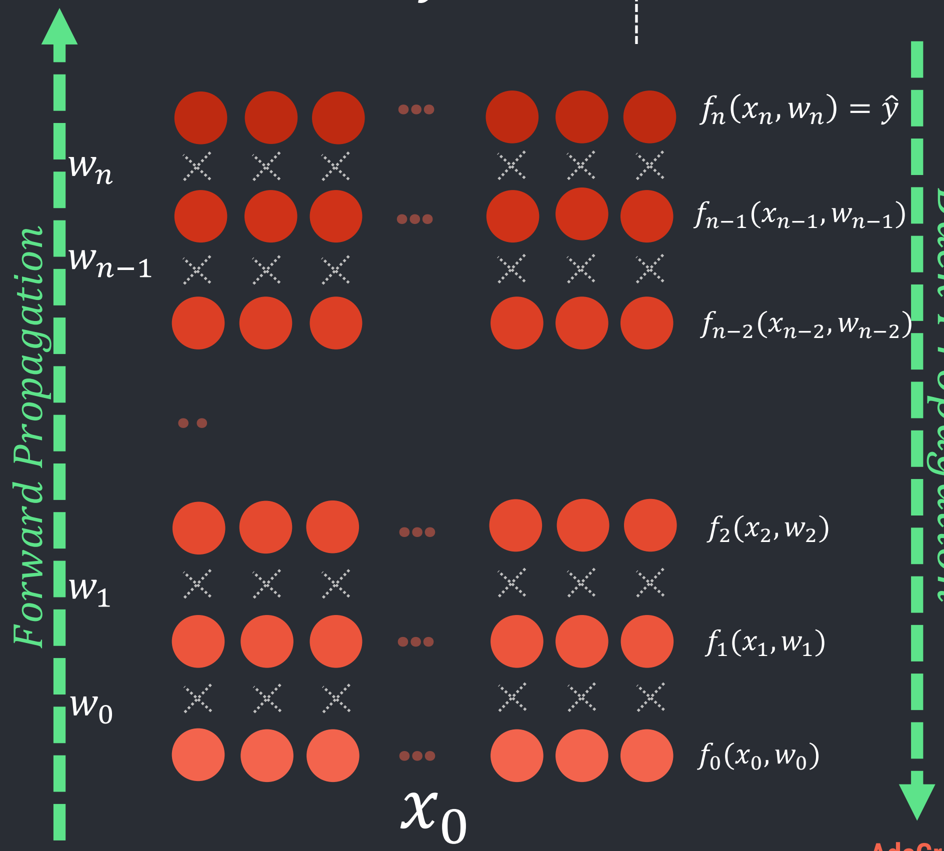
Update Rule

In [1]:

```
def model(X, weights):
    return X * weights

w = theano.shared(np.asarray(0., dtype=theano.config.floatX))
y = model(X, weights)

Loss = T.mean(T.sqr(y - Y))
gradient = T.grad(loss, weights)
updates = [[weights, weights - gradient * learning_rate]]

train = theano.function(inputs=[X, Y], outputs=loss, updates=updates,
allow_input_downcast=True)

for i in range(epoches):
    for x, y in zip(X, Y):
        train(x, y)
```

GRADIENT BASED MODELS

1: Forward Propagation | **2: Loss Calculation** | **3: Optimization**



$$E = l(\hat{y}, y)$$

$$\frac{\partial E}{\partial x_n} = \frac{\partial l(\hat{y}, y)}{\partial x_n}$$

$$\frac{\partial E}{\partial x_{n-1}} = \frac{\partial E}{\partial x_n} \frac{\partial f_n(x_{n-1}, w_n)}{x_{n-1}}$$

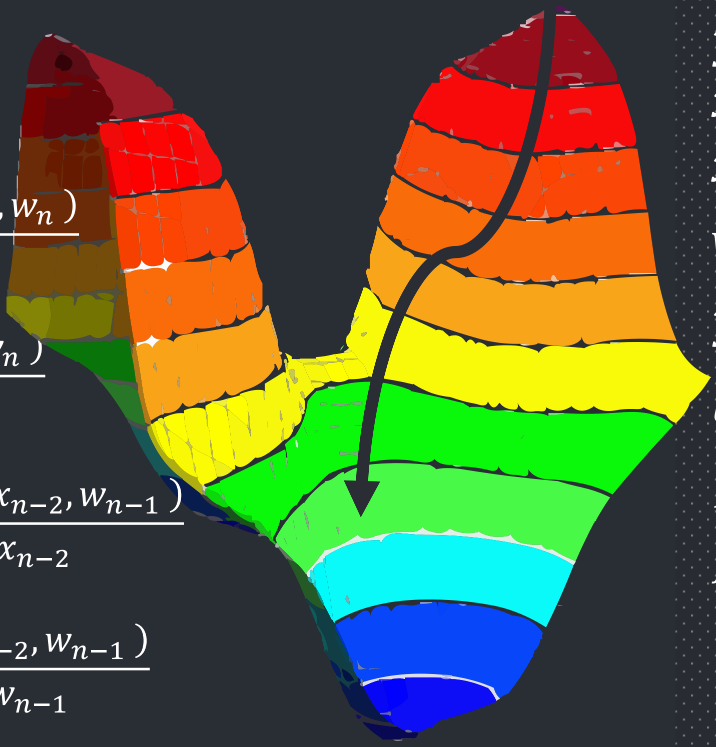
$$\frac{\partial E}{\partial w_n} = \frac{\partial E}{\partial x_n} \frac{\partial f_n(x_{n-1}, w_n)}{\partial w_n}$$

$$\frac{\partial E}{\partial x_{n-2}} = \frac{\partial E}{\partial x_{n-1}} \frac{\partial f_{n-1}(x_{n-2}, w_{n-1})}{x_{n-2}}$$

$$\frac{\partial E}{\partial w_{n-1}} = \frac{\partial E}{\partial x_{n-1}} \frac{\partial f_n(x_{n-2}, w_{n-1})}{\partial w_{n-1}}$$

...

...



Legend

- y – Ground Truth
- x_0 - Features Vector
- x_i - Output of i layer
- w_i - Weights of i layer
- $\hat{y}$ – Model Output
- $l(\hat{y}, y)$ - Loss Function
- E – Loss Surface
- f – Activation Function

Classic SGD

$$v_t = \mu v_{t-1} - \alpha \nabla L_t(w_{t-1})$$
$$w_t = w_{t-1} - V_t$$

AdaGrad

$$w_t = w_{t-1} - \alpha \frac{\nabla L_t(w_{t-1})}{\sqrt{\sum_{t'=1}^{t} \nabla L_{t'}(w_{t'-1})^2}}$$

RMSProp

$$R_t = \gamma R_{t-1} + (1-\gamma) \nabla L_t(w_{t-1})^2$$
$$w_t = w_{t-1} - \alpha \frac{\nabla L_t(w_{t-1})}{\sqrt{R_t}}$$

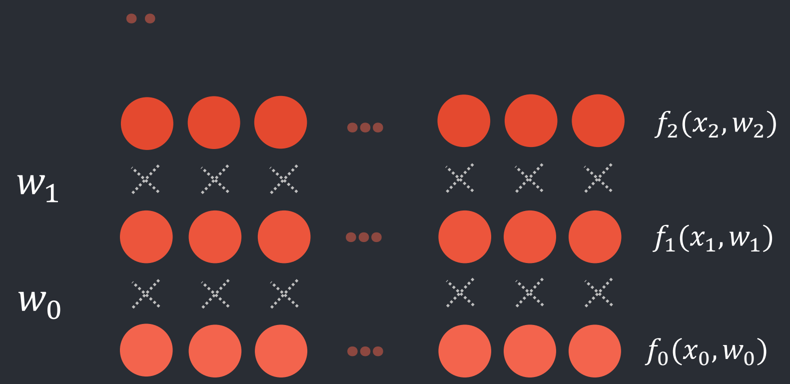
Adam

$$M_t = \frac{\beta_1 M_{t-1} + (1-\beta_1) \nabla L_t(w_{t-1})}{(1-\beta_1)^t}$$
$$R_t = \frac{\beta_2 M_{t-1} + (1-\frac{\beta_2}{2}) \frac{\nabla L_t(w_{t-1})}{2}}{(1-\beta_2)^2}$$
$$w_t = w_{t-1} - \alpha \frac{M_t}{\sqrt{R_t}}$$

GRADIENT BASED MODELS

1: Forward Propagation

```
def model(X, weights)
        …
        …
```



$$\dots f\left(f\left(f(x)\right)\right)\dots$$

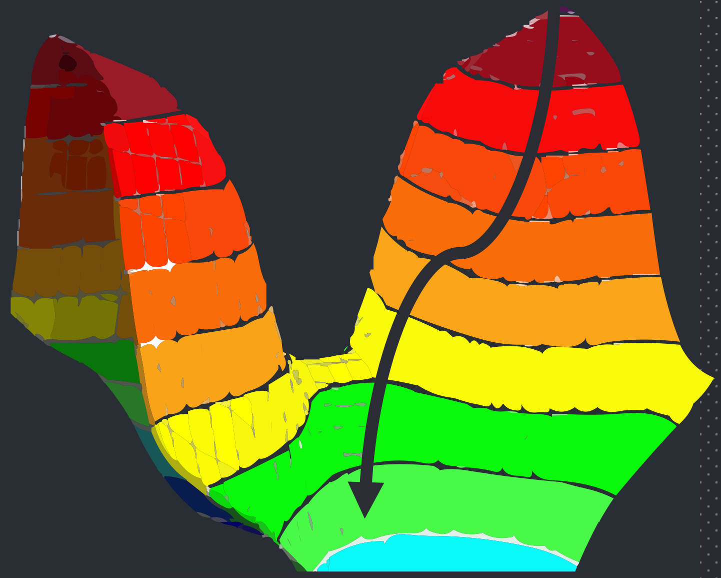
2: Loss Calculation

```
Loss = …
```

$$l(\hat{y}, y)$$

3: Optimization

```
gradient =
T.grad(loss, weights)

updates =
[[weights, weights - gradient]]
```



$$\frac{\partial E}{\partial x} = \frac{\partial l(\hat{y}, y)}{\partial x}$$

Legend

- y – Ground Truth
- x_0 - Features Vector
- x_i - Output of i layer
- w_i - Weights of i layer
- $\hat{y}$ – Model Output
- $l(\hat{y}, y)$ - Loss Function
- E – Loss Surface
- f – Activation Function

Keras Tutorial

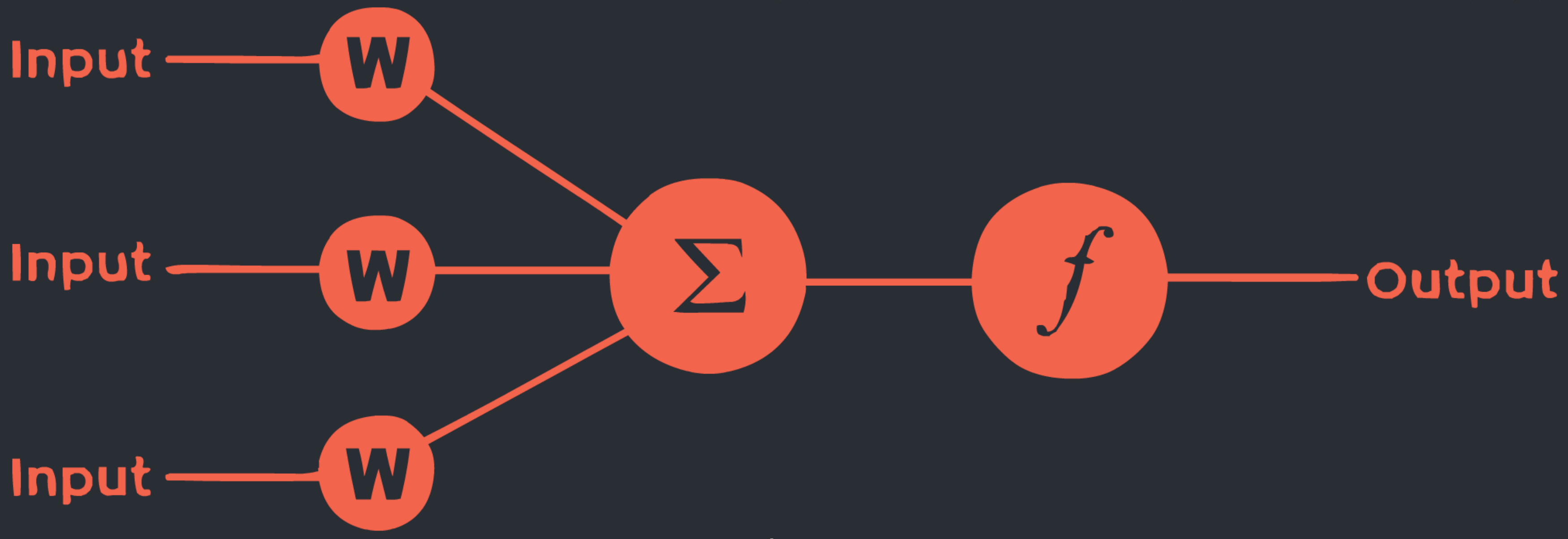
Simple Squential

Sequential

The core data structure of Keras is a model, a way to organize layers. The main type of model is the Sequential model, a linear stack of layers.

```
In [1]: from keras.models import Sequential
        from keras.layers.core import Dense, Activation

        model = Sequential()
```

```
model.add(Dense(output_dim=…, input_dim=…))    model.add(Activation(…))
```



Keras Tutorial

Simple Squential

Sequential

The core data structure of Keras is a model, a way to organize layers. The main type of model is the Sequential model, a linear stack of layers.

In [1]:
```
from keras.models import Sequential
from keras.layers.core import Dense, Activation

model = Sequential()
model.add(Dense(output_dim=64, input_dim=100))
model.add(Activation("relu"))
model.add(Dense(output_dim=24, input_dim=64))
model.add(Activation("relu"))
model.add(Dense(output_dim=10))
model.add(Activation("softmax"))

model.compile(loss='categorical_crossentropy', optimizer='sgd')
model.fit(X,Y)
```

Deep Neural Networks

For complex function approximation

Features

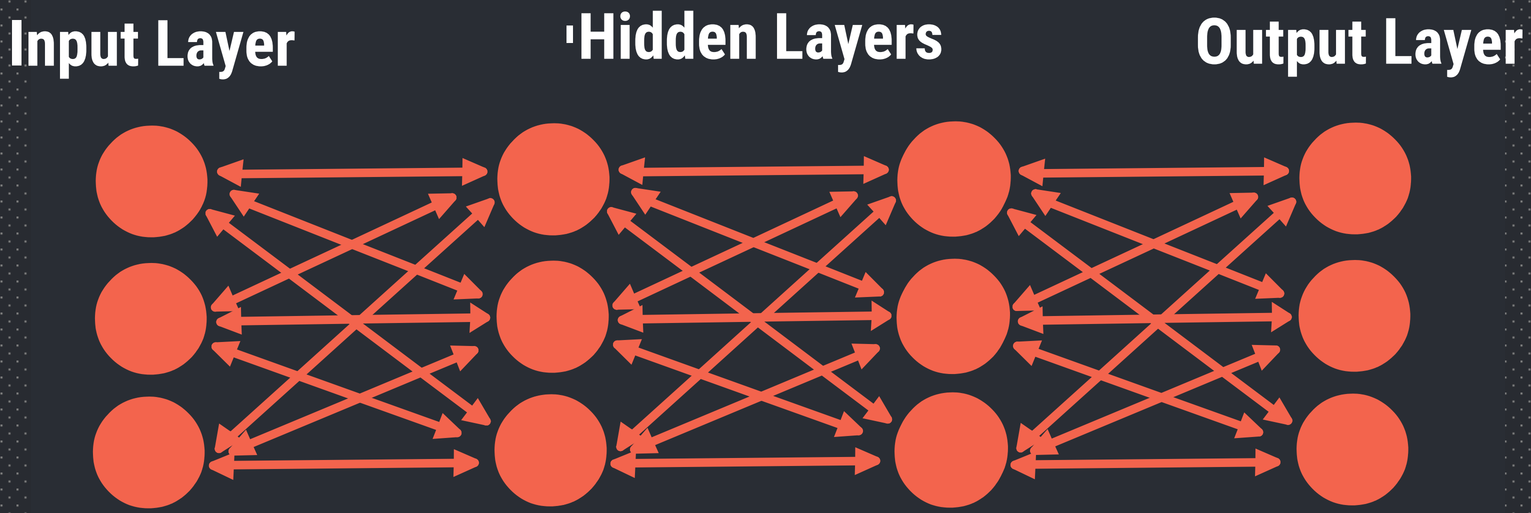
Past Prices

Correlations

Technical Analysis

Z Score

Time Features



Ground Truth

Future Prices
Regression

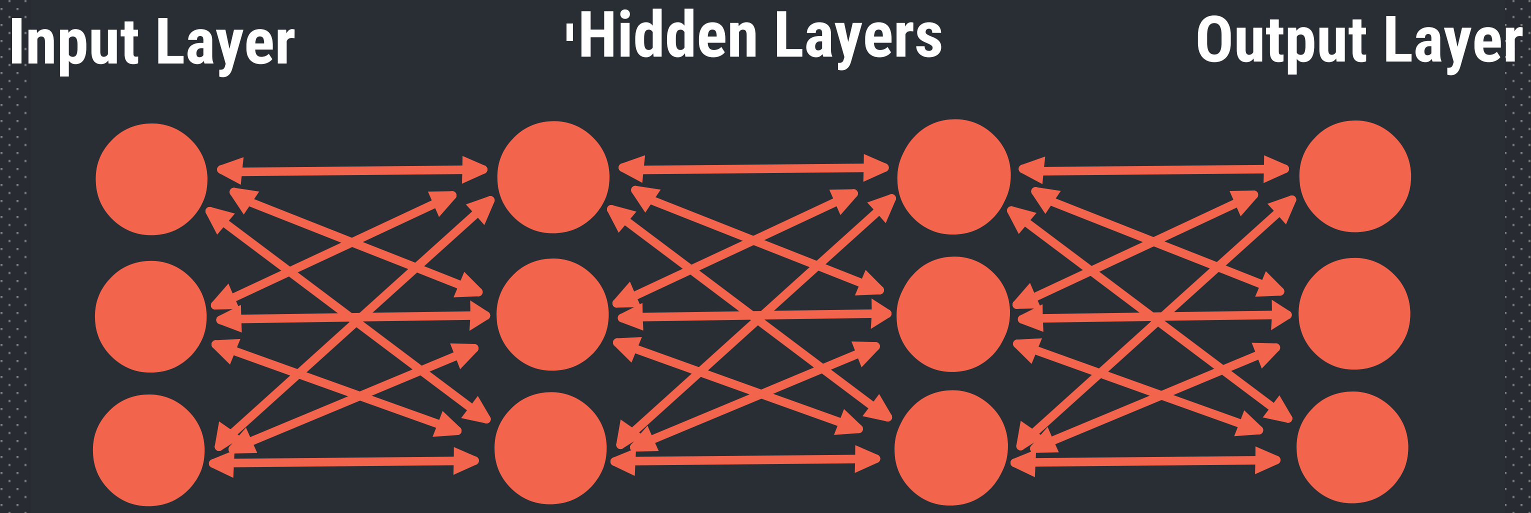
Up or Down
Classification

Recommended Papers

Implementing deep neural networks for financial market prediction, Dixon et al, 2015

Deep Neural Networks

For complex function approximation



Features

- Past Prices
- Correlations
- Technical Analysis
- Z Score
- Time Features

Ground Truth

- Future Prices
  Regression
- Up or Down
  Classification

```
model = Sequential()
model.add(Dense(output_dim=64, input_dim=100))
model.add(Activation("relu"))
model.add(Dense(output_dim=24, input_dim=64))
…
model.compile(loss='categorical_crossentropy',
                                   optimizer='sgd')
model.fit(X,Y)
```

Auto Encoders

For learning the distribution of the features

Features

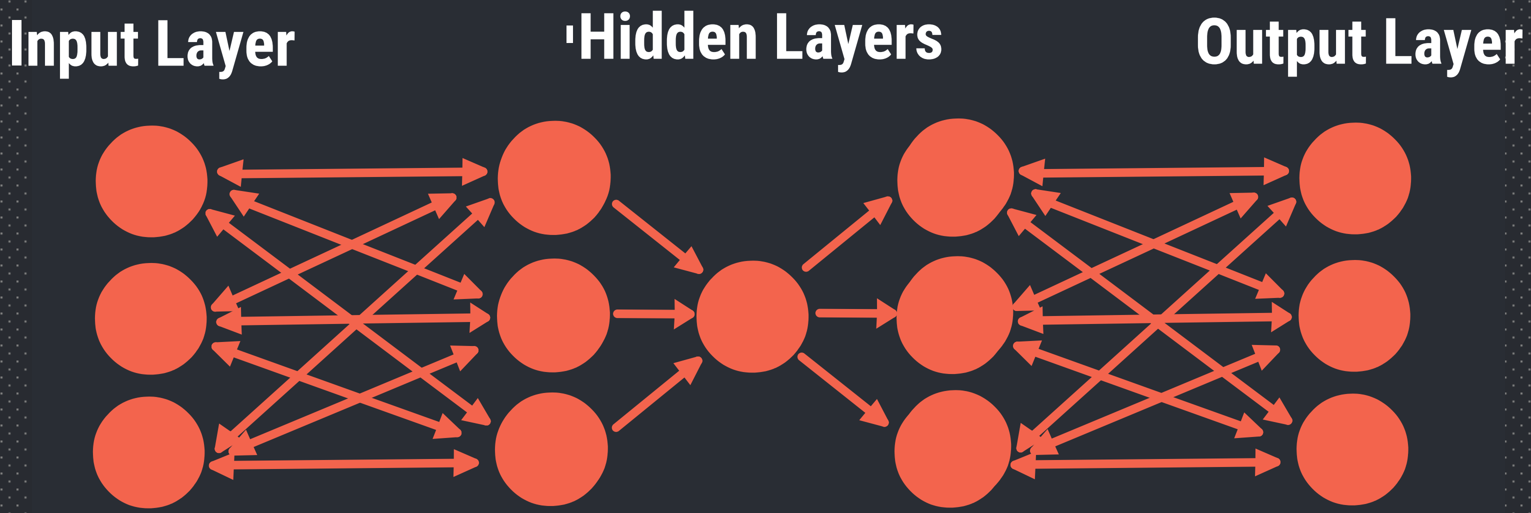
Past Prices

Correlations

Technical Analysis

Z Score

Time Features



Features

Past Prices

Correlations

Technical Analysis

Z Score

Time Features

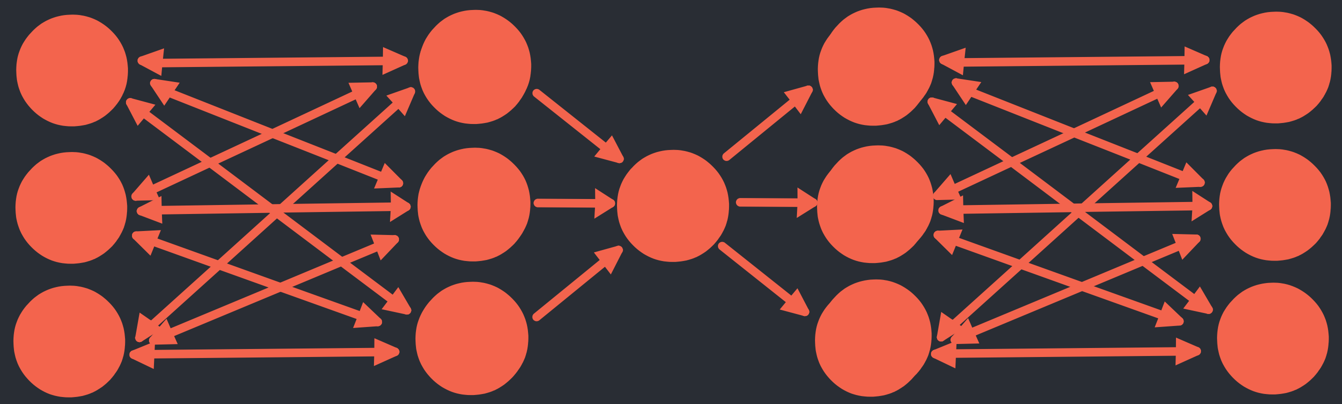
Recommended Papers

Deep Modeling Complex Couplings within Financial Markets, Cao et al, AAAI 2015

Auto Encoders

For learning the distribution of the features

Features	Input Layer	Hidden Layers	Output Layer	Features



Features
Past Prices
Correlations
Technical Analysis
Z Score
Time Features

```
model = Sequential()
model.add(Dense(output_dim=64, input_dim=100))
model.add(Activation("relu"))
model.add(Dense(output_dim=24, input_dim=64))
…
model.compile(loss='categorical_crossentropy',
                                            optimizer='sgd')
model.fit(X,X)
```

Features
Past Prices
Correlations
Technical Analysis
Z Score
Time Features

Questions?