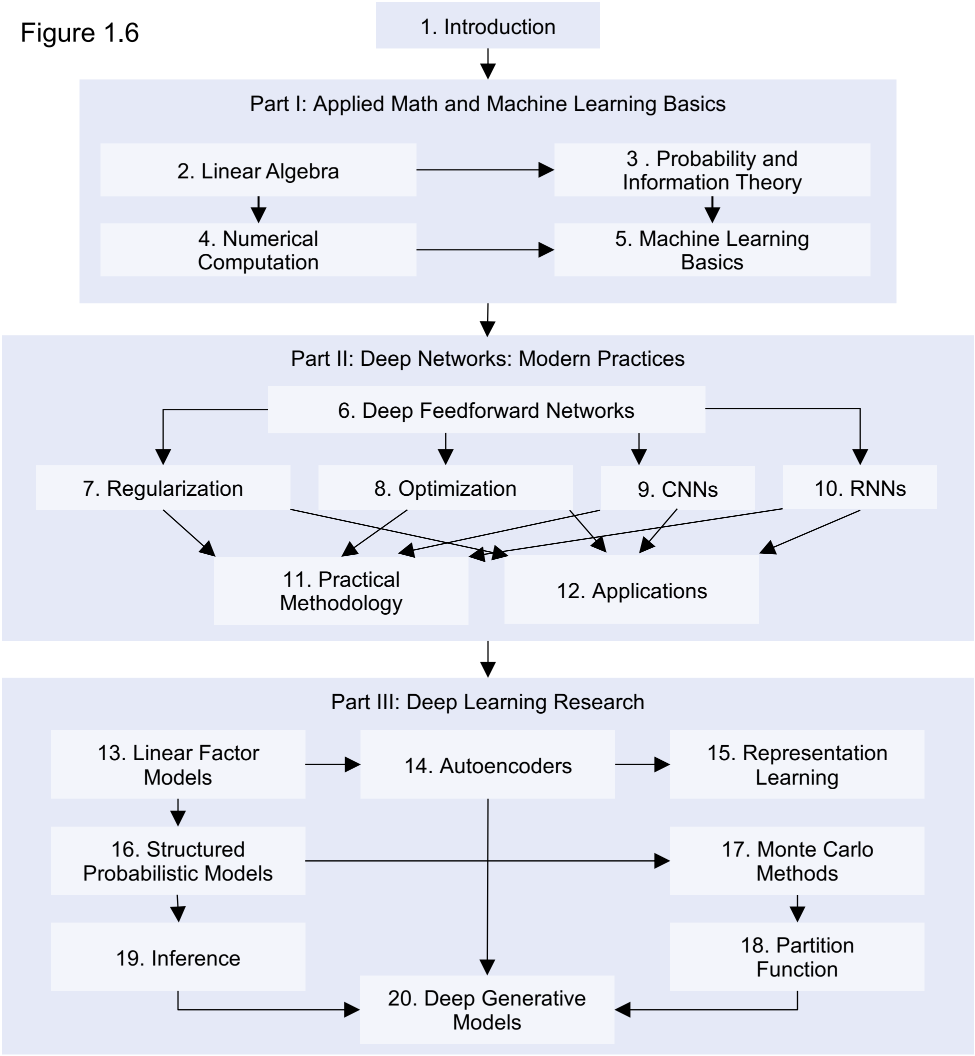
Neural Networks

Learning a Transformation Matrix

Organization of the Book

Figure 1.6



(Goodfellow 2016)

Source: *Deep Learning*,(http://www.deeplearningbook.org/slides/01_intro.pdf)

DataScience@SMU

Perceptron

Perceptron

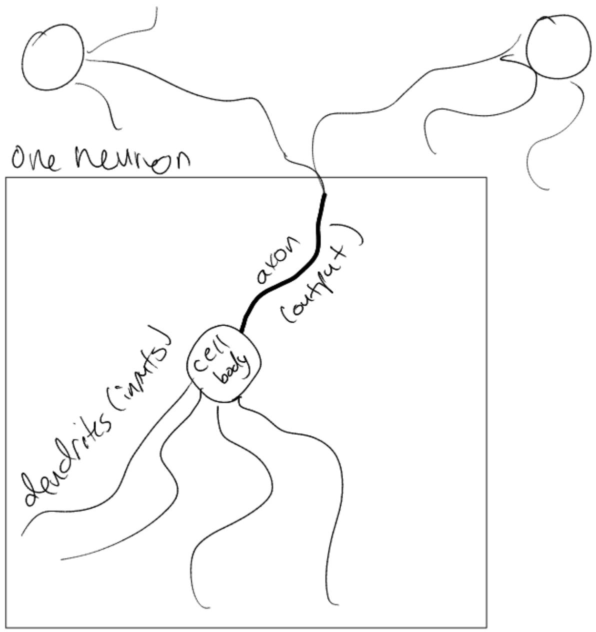


Figure 4.1: A picture of a neuron

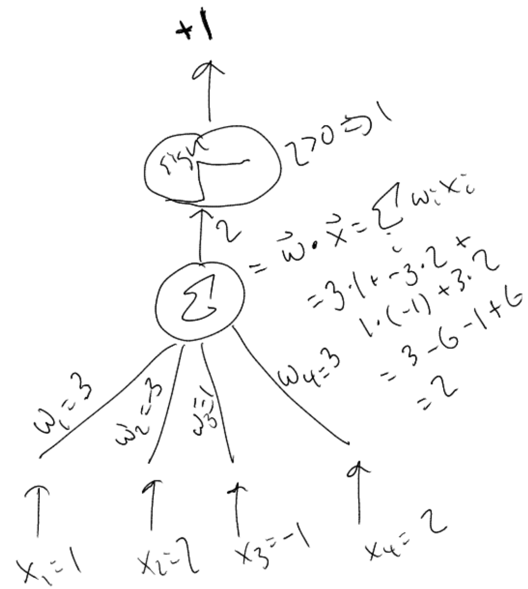


Figure 4.2: Figure showing feature vector and weight vector and products and sum

Perceptron

- The perceptron converges when the data is linearly separable.
- Nonlinear decision boundary: combining multiple perceptrons in a single framework; i.e., neural networks.

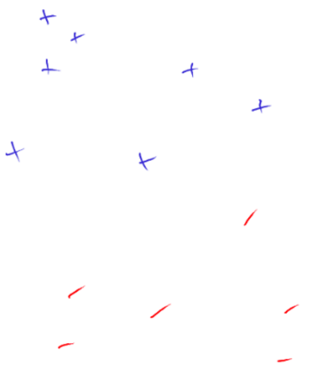


Figure 4.10: Separable data

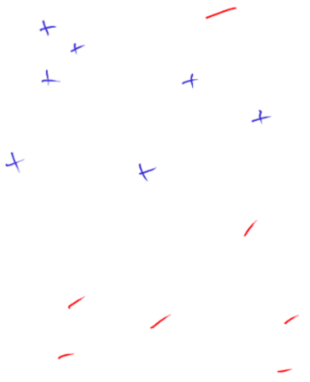


Figure 4.11: Inseparable data

Source: *A Course in Machine Learning*, Ha Daume III (http://ciml.info/)

Perceptron, Infinitely Wide

Chaining together perceptrons to build more complex neural networks such as:

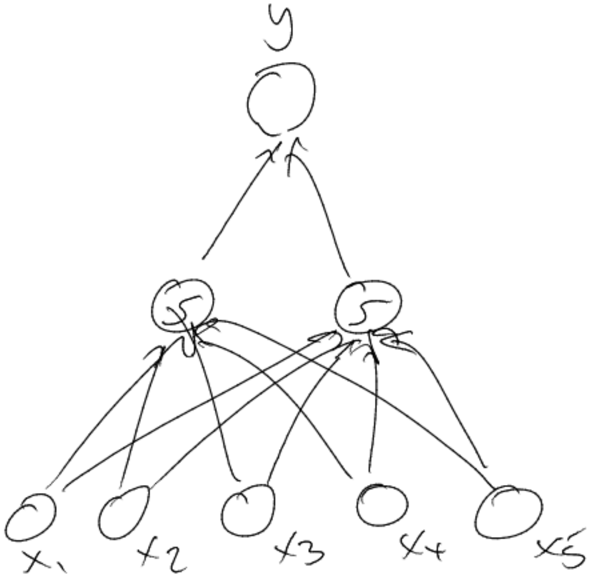


Figure 10.1: Picture of a two-layer network with five inputs and two hidden units

Five features, two layers (two hidden united and one output unit), the edges are called weights (learned during training).

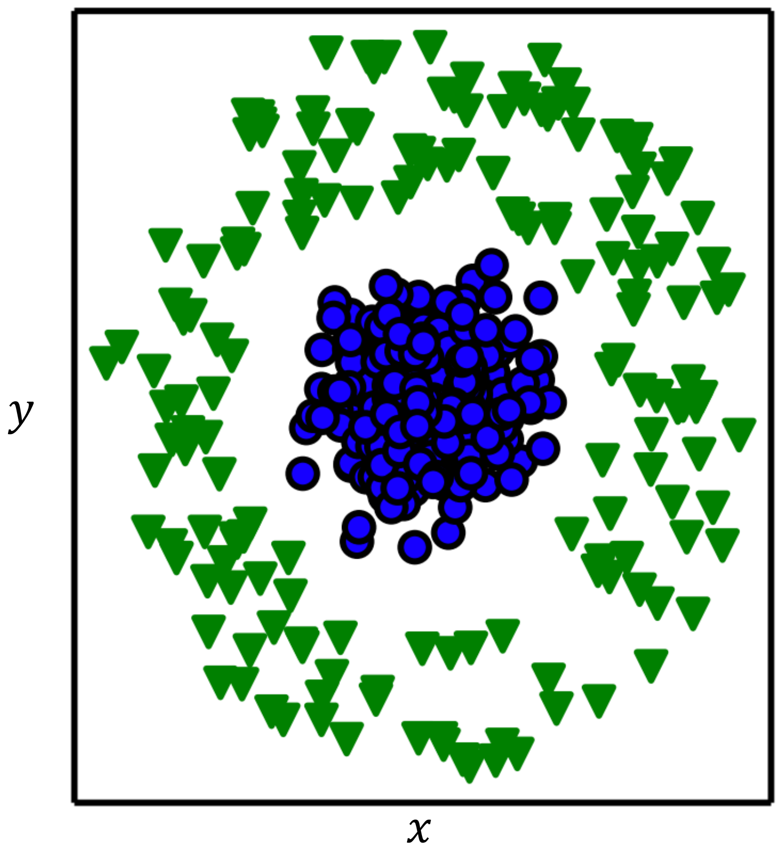
Source: *A Course in Machine Learning*, Ha Daume III (http://ciml.info/)

DataScience@SMU

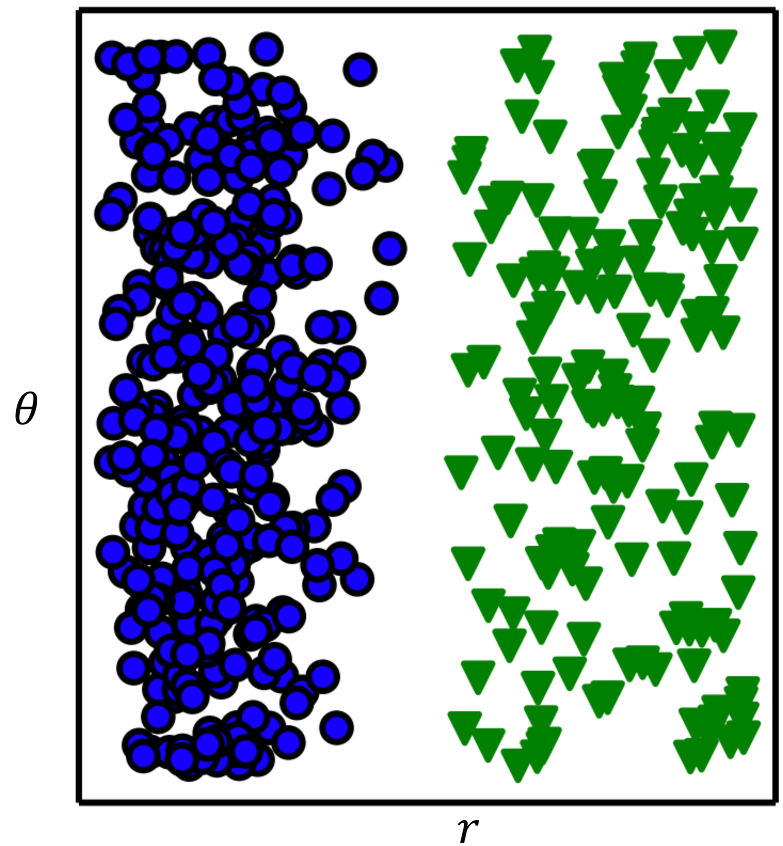
Representations

Representations Matter

Cartesian coordinates



Polar coordinates

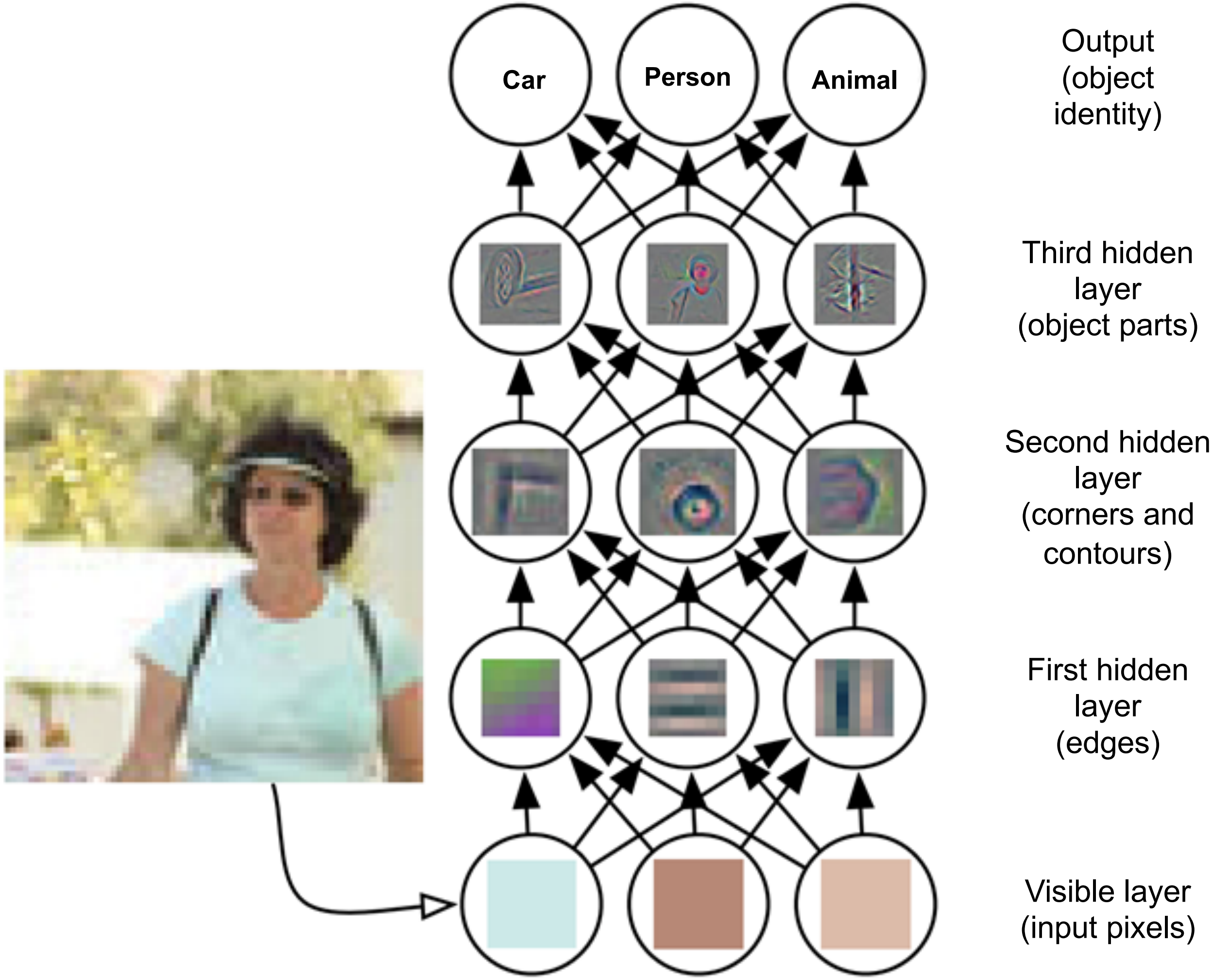


Source: *Deep Learning*,(http://www.deeplearningbook.org/slides/01_intro.pdf)

DataScience@SMU

Understanding Depth

Depth: Repeated Composition



Source: *Deep Learning*,(http://www.deeplearningbook.org/slides/01_intro.pdf)

DataScience@SMU

Keras Example

```
import sklearn.datasets as datasets

from keras.models import Sequential
from keras.layers import Dense
from keras.utils.np_utils import to_categorical

M = datasets.load_iris()['data']
L = to_categorical(datasets.load_iris()['target’])

n_cols = M.shape[1]
n_labels = L.shape[1]

model = Sequential()
model.add(Dense(25, input_dim=n_cols, activation='relu'))
model.add(Dense(n_labels, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy’])

model.fit(M, L, batch_size=10, epochs=25)
```

DataScience@SMU

Python Examples

```
X = np.array([ [0,0,1],[0,1,1],[1,0,1],[1,1,1] ])
y = np.array([[0,1,1,0]]).T
syn0 = 2*np.random.random((3,4)) - 1
syn1 = 2*np.random.random((4,1)) - 1
for j in range(60000):
    l1 = 1/(1+np.exp(-(np.dot(X,syn0))))
    l2 = 1/(1+np.exp(-(np.dot(l1,syn1))))
    l2_delta = (y - l2)*(l2*(1-l2))
    l1_delta = l2_delta.dot(syn1.T) * (l1 * (1-l1))
    syn1 += l1.T.dot(l2_delta)
    syn0 += X.T.dot(l1_delta)
```

https://iamtrask.github.io/2015/07/12/basic-python-network/

```
import numpy as np
np.random.seed(1)

def nonlin(x,deriv=False):
    if(deriv==True):
        return x*(1-x)
    return 1/(1+np.exp(-x))

X = np.array([[0,0,1], [0,1,1], [1,0,1], [1,1,1]])

y = np.array([[0,0,1,1]]).T
syn0 = 2*np.random.random((3,1)) - 1

for iter in range(10000):
    # forward propagation
    l0 = X
    l1 = nonlin(np.dot(l0,syn0))
    # how much did we miss?
    l1_error = y - l1
    # multiply how much we missed by the
    # slope of the sigmoid at the values in l1
    l1_delta = l1_error * nonlin(l1,True)
    # update weights
    syn0 += np.dot(l0.T,l1_delta)

print (l1)
```

References

Figures source: *A Course in Machine Learning*, Ha Daume III (http://ciml.info/)

DataScience@SMU