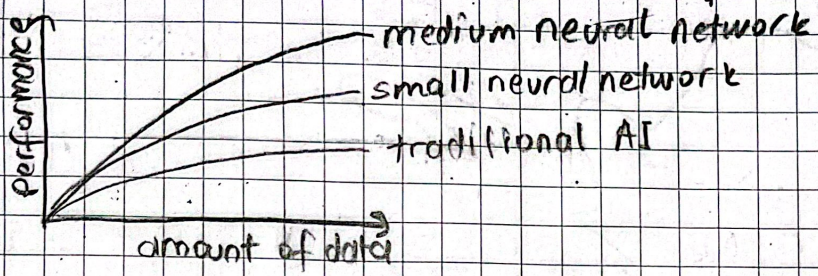
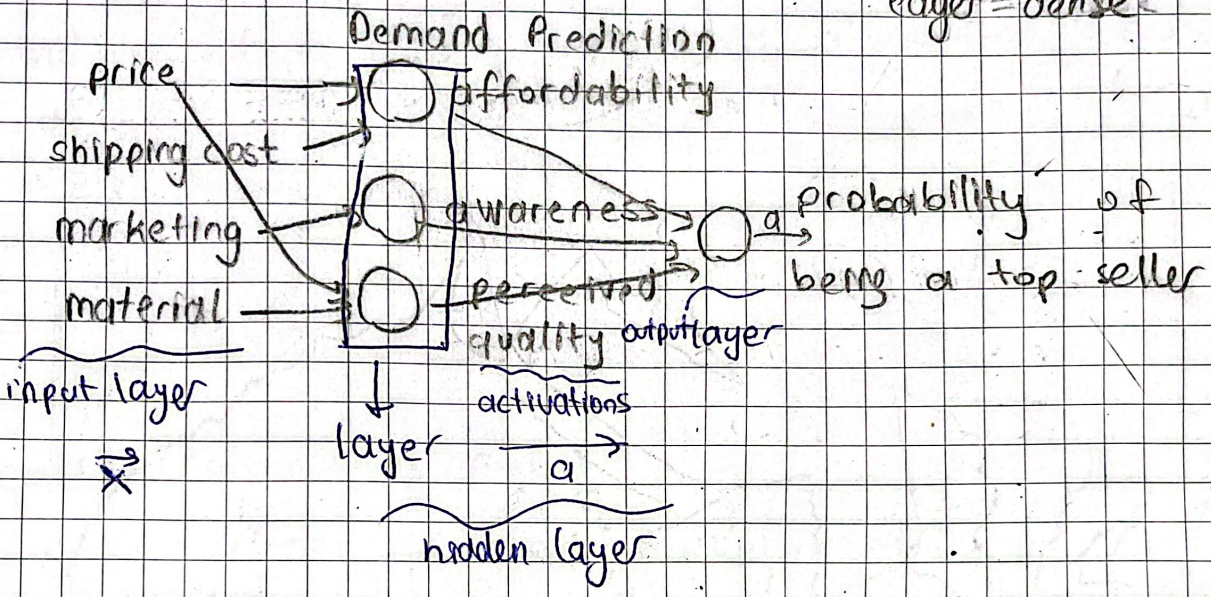
Neural Networks

Speech → images → text (NLP)

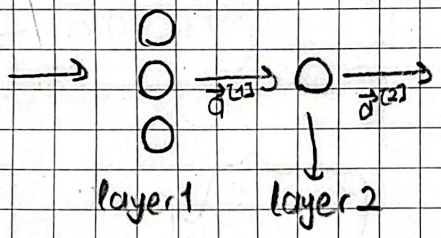


layer = dense



multilayer perceptron

Neural network layer



Inference: making predictions (forward propagation)

Tensorflow implementation

x = np.array([[200,17]]) [200,17] 1x2

x = np.array([200,17]) 1D vector

```
x=np.array([[200.0, 17.0]])
layer_1= Dense(units=3, activation="sigmoid")
a1= layer_1(x)
layer_2=Dense(units=1, activation="sigmoid")
a2= layer_2(a1)
```

forward propagation

AI

ANI self-driving car

AGI Do anything a human can do

```
def dense(A_in,W,B):
    Z = np.matmul(A_in, W)+B
    A_out = g(Z)
    return A_out
AT= A.T
Z = np.matmul(AT,W) = AT@W
```

epochs: number of steps in gradient descent

neural network

```
model = Sequential([
                    Dense(...)
                    Dense(...)
                    ])
```

binary cross entropy

```
model.compile(loss= BinaryCrossentropy())
model.fit(X,y, epochs=100)
```

Examples of Activation Functions

Linear activation function (NO activation function)

Sigmoid

ReLU (Rectified Linear Unit)

Softmax regression alghoritm

$\times\ z_1 = \vec{w}_1 \cdot \vec{x} + b_1$
$\bigcirc\ z_2 = \vec{w}_2 \cdot \vec{x} + b_2$
$\square\ z_3 = \vec{w}_3 \cdot \vec{x} + b_3$
$\triangle\ z_4 = \vec{w}_4 \cdot \vec{x} + b_4$

$$a_1 = \frac{e^{z_1}}{e^{z_1} + e^{z_2} + e^{z_3} + e^{z_4}} \quad (\times\ \bigcirc\ \square\ \triangle)$$

$$= P(y = 1 | \vec{x})$$

Adam alghoritm Intuition

Adam: (Ada)ptive (M)oment estimation not just one α

Additional layer types

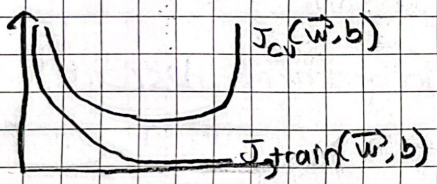
- convolutional layer

Train/test procedure for linear regression (with squared error cost)

Fit parameters by minimizing cost function $J(\vec{w}, b)$

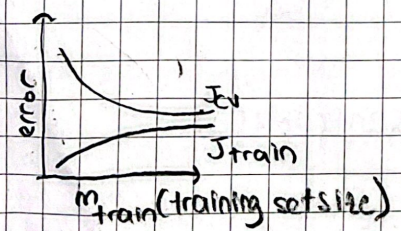
$$J(\vec{w}, b) = \frac{1}{2 m_{train}} \sum_{i=1}^{m_{train}} \left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right)^2 + \frac{\lambda}{2 m_{train}} \sum_{j=1}^{n} w_j^2$$

Diagnosing bias and variance



learning curves

J_{train} = training error J_{cv} = cross validation error



Debugging a learning algorithm

You've implemented regularized linear regression on housing prices but still large errors.

Get more training example → fixes high variance

Try smaller sets of features → high variance

Try getting additional features → high bias

Try adding polynomial features → fixes high bias

Try decreasing λ fix high bias

Try increasing λ fix high variance

Large neural networks are low bias machines.

Conventional model-centric approach: AI = Code + Data (work on this)

Data-centric approach: AI = Code + Data (work on this)

Transfer learning

Decision Trees

$p_0 = 1 - p_1$

$H(p_1) = -p_1 \log_2(p_1) - p_0 \log_2(p_0)$ (+1/2, ①, 2^{-1})

$0 \log(0) = 0$

One-hot encoding

Tree ensemble

sampling with replacement

Random forest

XGBoost (eXtreme Gradient Boosting)

Decision trees vs Neural Networks