Deep NN Learning
Problem Types, Last-layer Activation, and Loss Functions

Problem type	Last-layer activation	Loss function
Binary classification	`sigmoid`	`binary_crossentropy`
Multiclass, single-label classification	`softmax`	`categorical_crossentropy`
Multiclass, multilabel classification	`sigmoid`	`binary_crossentropy`
Regression to arbitrary values	None	`mse` or `mae`
Regression to values between 0 and 1	`sigmoid`	`mse` or `binary_crossentropy`

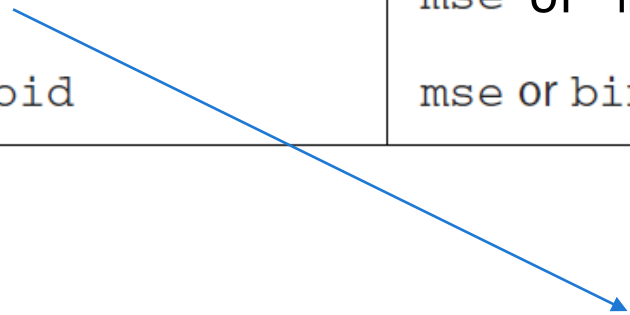
ML/DL Problem Types using Supervised Learning

- Binary Classification
- Multi class **single-label** classification
- Multi class **multi-label** classification

- Scala regression
  - Regression to arbitrary values
  - Regression to values between 0 and 1 (logistic regression, binary classification)
- Vector regression (a set of continuous values)

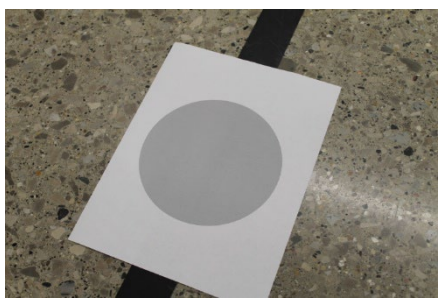
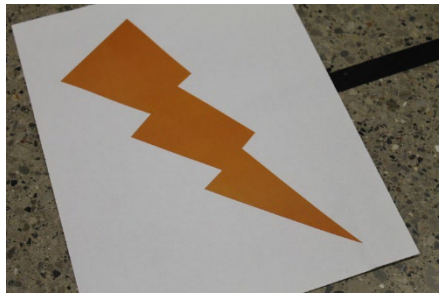
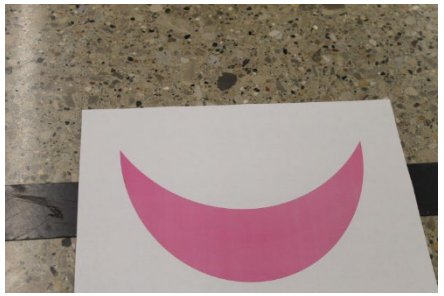
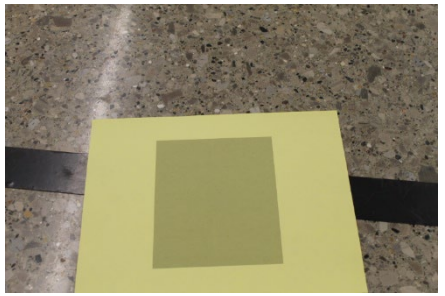
Choosing the right last-layer activation and loss function for your model

Problem type	Last-layer activation	Loss function
Binary classification	`sigmoid`	`binary_crossentropy`
Multiclass, single-label classification	`softmax`	`categorical_crossentropy`
Multiclass, multilabel classification	`sigmoid`	`binary_crossentropy`
Regression to arbitrary values	None	`mse` or `mae`
Regression to values between 0 and 1	`sigmoid`	`mse` or `binary_crossentropy`

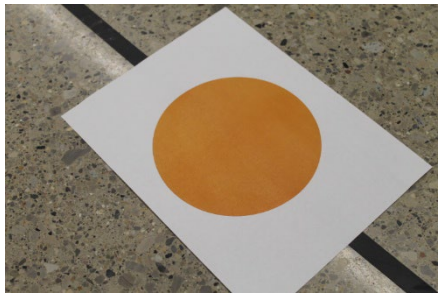
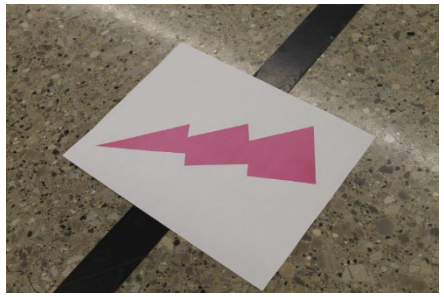
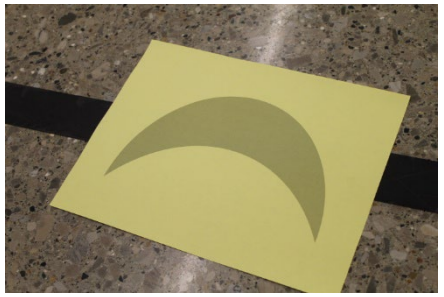
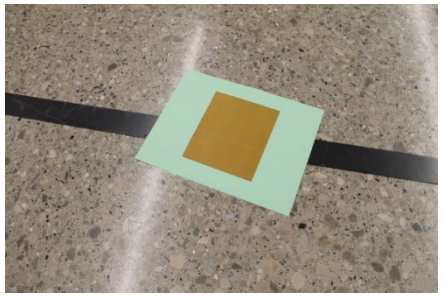


Like linear function

Multi class single-label classification (Robofest 2015 Vcc Problem)

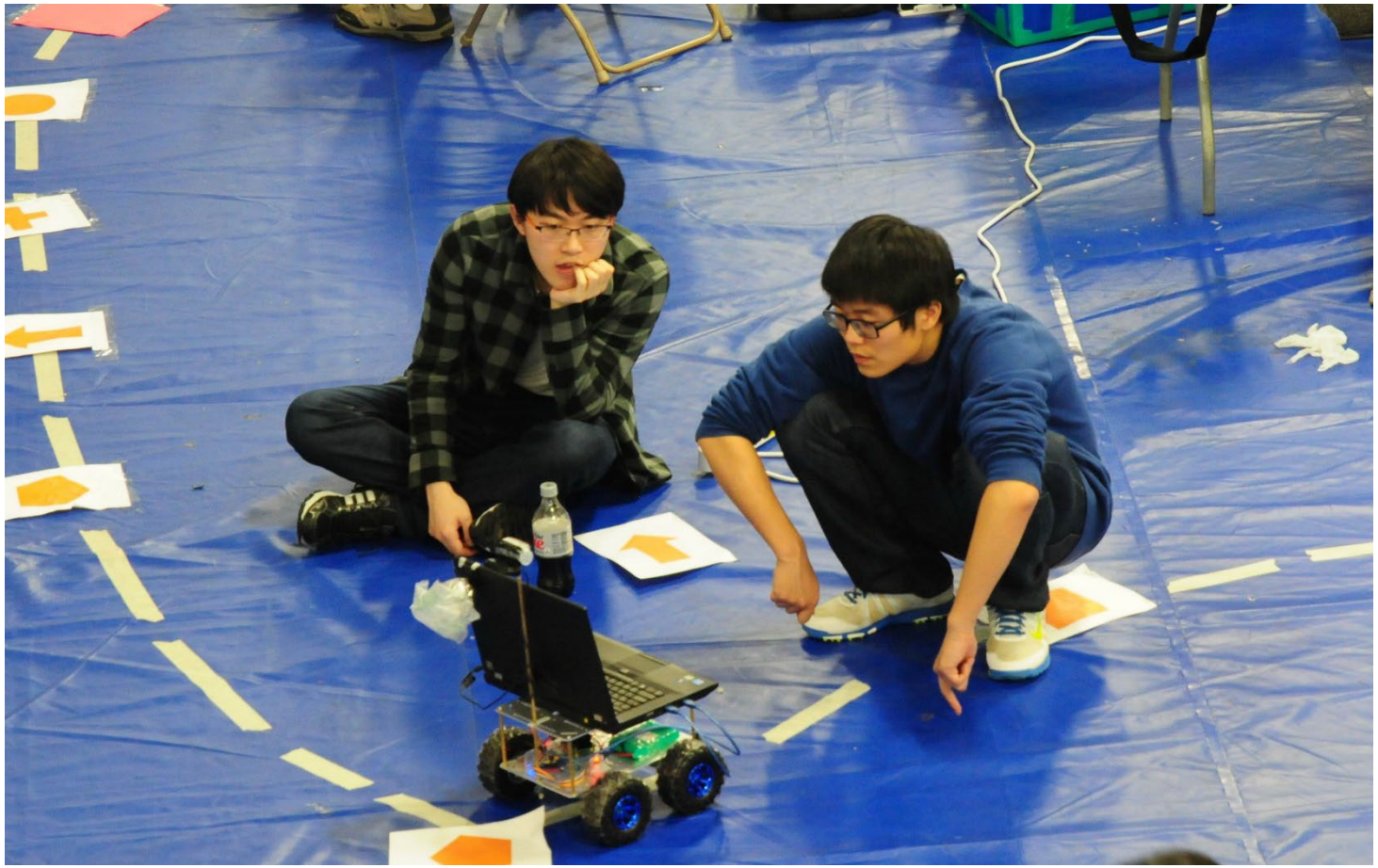





. . . .

[1 0 0 0] [0 1 0 0] [0 0 1 0] [0 0 0 1]

2015 Robofest Vcc



College Waypoint Numeric Value Chart	
Shape	***Numeric Value***
Circle	0
Square	1
Triangle	2
Pentagon	3
Hexagon	4
Cross	5
Arrow	6
Moon	7
Heart	8
Star	9

https://youtu.be/jC85QHiKNxs?si=SI7gMSdwj6cZmHJG

Classify_Newswires.ipynb (in Canvas)

- Chollet book Section 4.2 (page 106)
- Multi-class single-label classification – a news wire is classified into **only one** category out of 46.

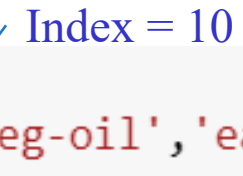
```
decoded_newswire
```

```
'? the european commission confirmed it authorised the export of 60 500 tonnes of current series wh
ite sugar at a maximum rebate of 43 147 european currency units ecus per 100 kilos out of this trad
ers in the u k received 43 500 tonnes in the netherlands 12 000 in denmark 4 000 and in west german
y 1 000 tonnes reuter 3'
```

10, sugar

Classify_Newswires.ipynb (in Canvas)

- Multi-class single-label classification – a news wire is classified into **only one** category out of 46.

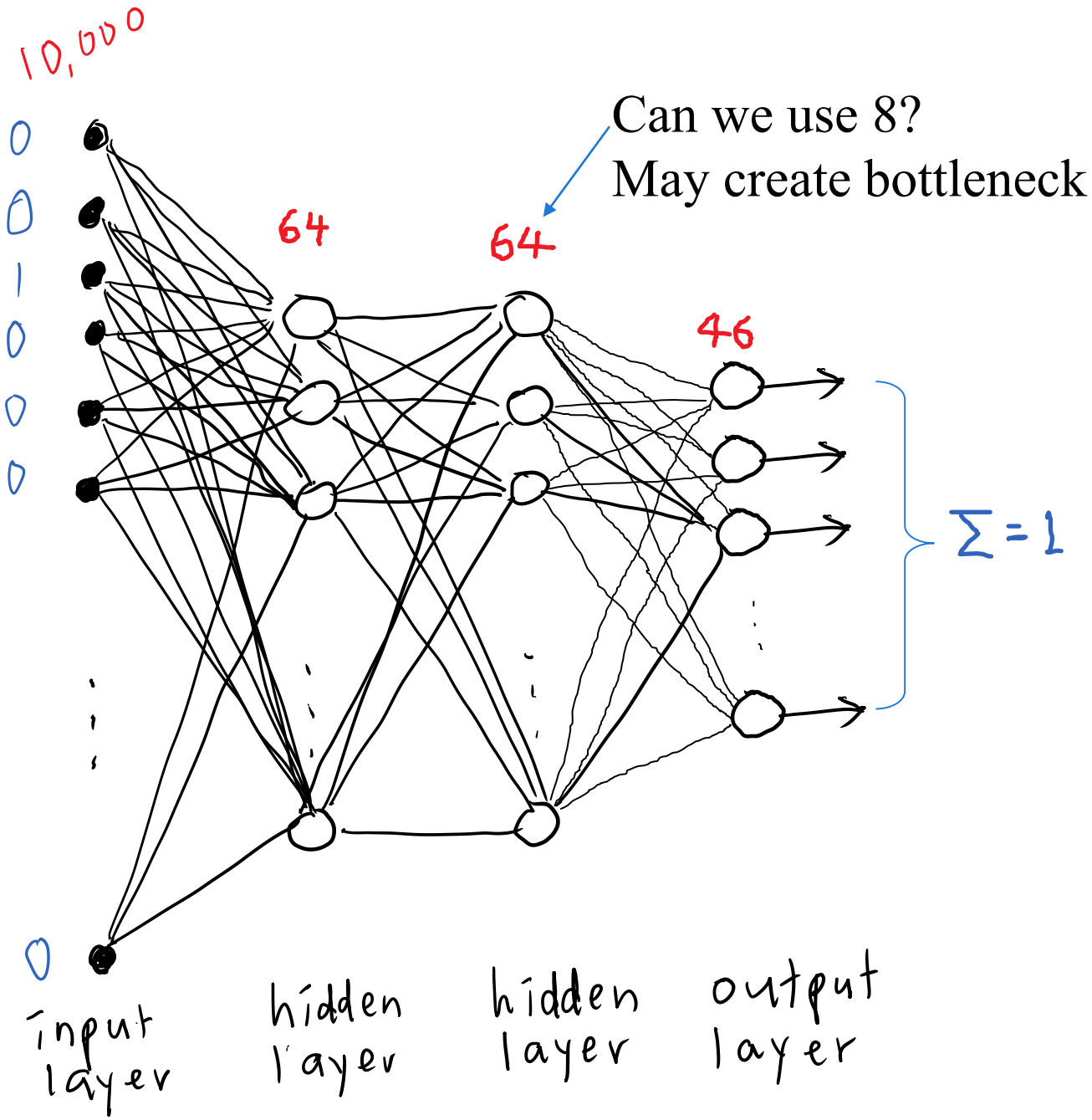



```
reuters46 =
['cocoa','grain','veg-oil','earn','acq','wheat','copper','housing','money-supply',
 'coffee','sugar','trade','reserves','ship','cotton','carcass','crude','nat-gas',
 'cpi','money-fx','interest','gnp','meal-feed','alum','oilseed','gold','tin',
 'strategic-metal','livestock','retail','ipi','iron-steel','rubber','heat','jobs',
 'lei','bop','zinc','orange','pet-chem','dlr','gas','silver','wpi','hog','lead']
```

- Input: same as IMDB movie review dataset.
  - Num_words=10000 restricts the data to the 10,000 frequently occurring words.
  - The function vectorize_sequences is used to generate vectors with 0s and 1s.

Review Questions

- ***Draw*** the network model for the Newswires
- How many input neurons?
- How many output neurons?
- How many hidden layers and hidden neurons?
- How many parameters to optimize?



The Model Used

```
model = keras.Sequential([
    layers.Dense(64, activation="relu"),
    layers.Dense(64, activation="relu"),
    layers.Dense(46, activation="softmax")
])
```

How many parameters?

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 64)	640064
dense_1 (Dense)	(None, 64)	4160
dense_2 (Dense)	(None, 46)	2990

Total params: 647,214
Trainable params: 647,214
Non-trainable params: 0

What is the meaning of the "None" in model.summary() of Keras?

- "None" means this dimension is variable
- The first dimension in a Keras model is always the batch size defined in `fit()` method
- The batch size is often ignored when you define your model. For instance, when you define input_shape = (100, 200), actually you're ignoring the batch size and defining the shape of "each sample". Internally the shape will be (None, 100, 200), allowing a variable batch size, each sample in the batch having the shape (100, 200).

```
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 dense (Dense)                (None, 64)                640064

 dense_1 (Dense)              (None, 64)                4160

 dense_2 (Dense)              (None, 46)                2990

=================================================================
Total params: 647,214
Trainable params: 647,214
Non-trainable params: 0
```

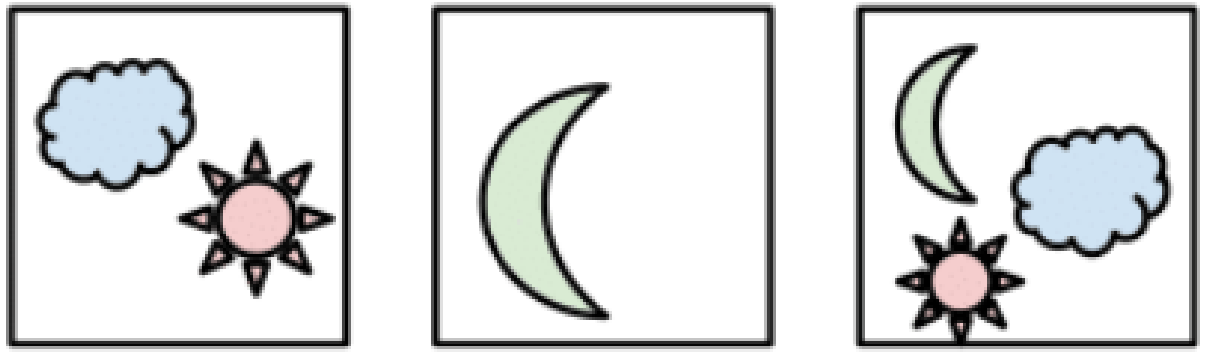
https://stackoverflow.com/questions/47240348/what-is-the-meaning-of-the-none-in-model-summary-of-keras

Compile and Train

```
model.compile(optimizer="rmsprop",
              loss="categorical_crossentropy",
              metrics=["accuracy"])
model.fit(x_train,
          y_train,
          epochs=9,
          batch_size=512)
results = model.evaluate(x_test, y_test)
```

Multi class **multi-label** classification, or simply **multi-label** classification

3 classes



[1 0 1] [0 1 0] [1 1 1]

Last layer activation: **sigmod**
Loss function: **binary_crossentropy**

Multi-label classification example

- **to predict both *color* and *clothing* type.**
- https://pyimagesearch.com/2018/05/07/multi-label-classification-with-keras/

100 100 010 010 010 100

010 001 001 010 001 001

BLK → 1

blue → 0

red → 0

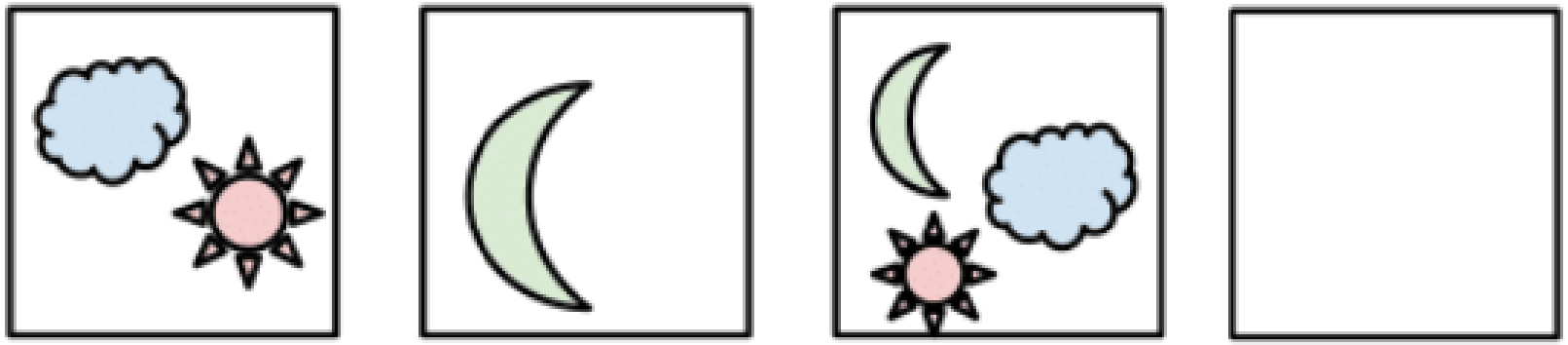
J → 1

D → 0

S → 0

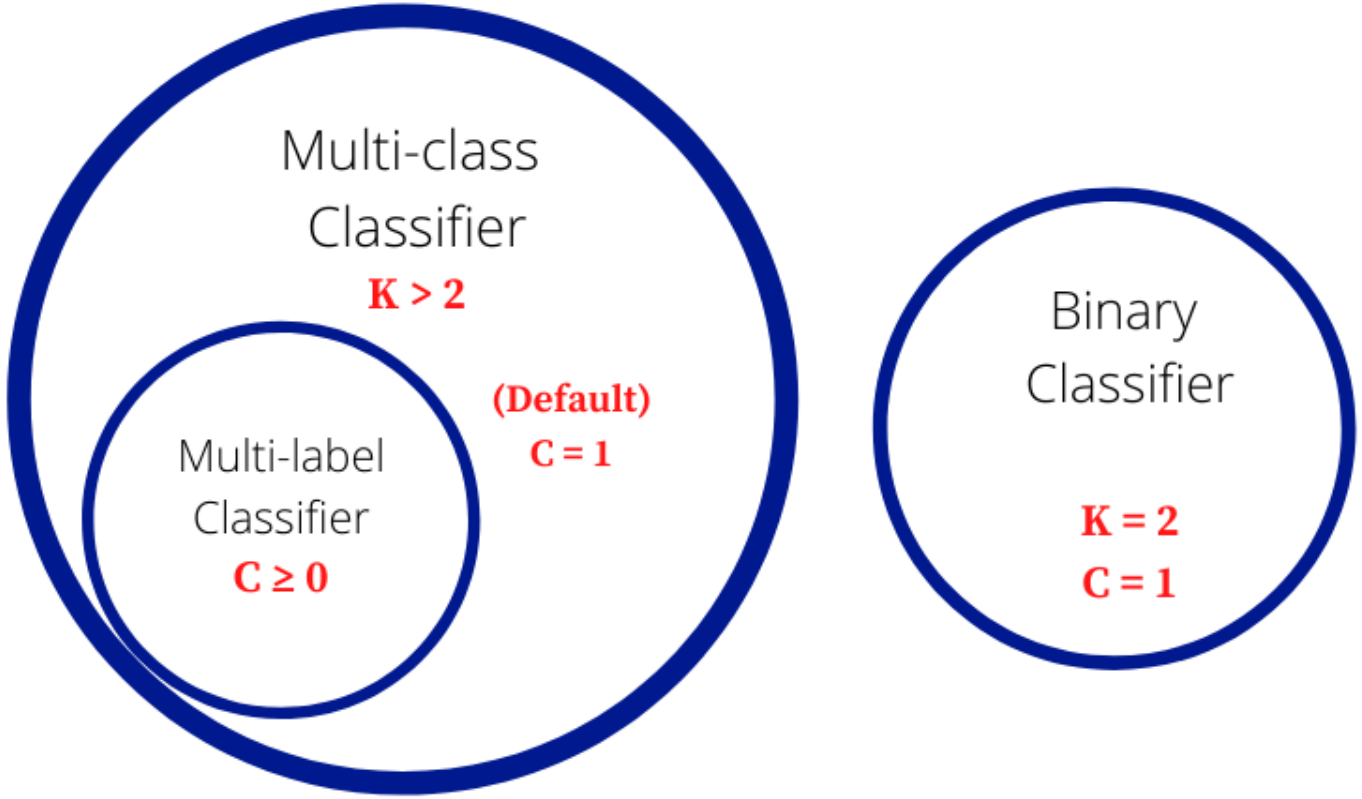
Multi class **multi-label** classification, or simply **multi-label** classification

3 classes



[1 0 1] [0 1 0] [1 1 1] ???

Last layer activation: sigmod
Loss function: binary_crossentropy



K = Total number of classes in the problem statement

C = Number of classes an item maybe assigned to

Loss Function

Problem type	Last-layer activation	Loss function
Binary classification	`sigmoid`	`binary_crossentropy`
Multiclass, single-label classification	`softmax`	`categorical_crossentropy`
Multiclass, multilabel classification	`sigmoid`	`binary_crossentropy`
Regression to arbitrary values	None	`mse` or mae
Regression to values between 0 and 1	`sigmoid`	`mse` or `binary_crossentropy`

Loss Function

- loss is a number indicating how bad the model's prediction was.
- A loss function (or Error function, objective function, or optimization score function) is one of the two parameters required to compile a model
- How the network will be able to measure its performance on the training data, and thus how it will be able to steer itself in the right direction
- Loss Functions used so far
  - mse: mean_squared_error (good for regression)
  - categorical_crossentroypy (good for classification)
- https://keras.io/losses/
- How to choose the right loss for fitting a model - https://heartbeat.fritz.ai/5-regression-loss-functions-all-machine-learners-should-know-4fb140e9d4b0

metrics and loss in model.compile ()

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

- A metric is **a function that is used to judge the performance of your model**.
- Metric functions are similar to loss functions, except that the results from evaluating a metric are *not used when training the model*.
- Note that you may use any loss function as a metric.
- Keras metric functions: https://keras.io/api/metrics/

Review: popular loss functions

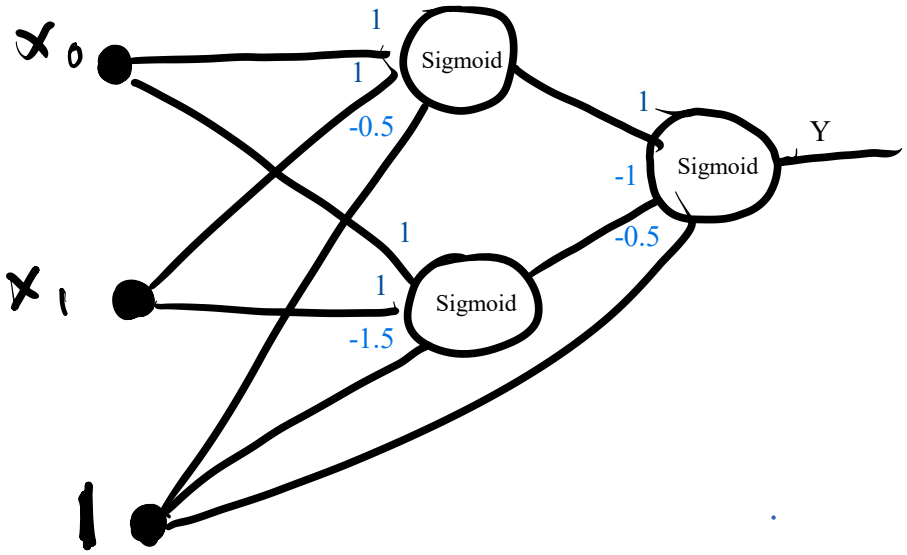
Mean squared error	$\text{MSE} = \frac{1}{n}\sum_{t=1}^{n} e_t^2$
Root mean squared error	$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{t=1}^{n} e_t^2}$
Mean absolute error	$\text{MAE} = \frac{1}{n}\sum_{t=1}^{n} \lvert e_t \rvert$
Mean absolute percentage error	$\text{MAPE} = \frac{100\%}{n}\sum_{t=1}^{n} \left\lvert \frac{e_t}{y_t} \right\rvert$

Self-Exercise 1: Write a simple Python code to calculate MSE, RMSE, MAE, and MAPE

- Input data examples and expected outputs:

X_0	X_1	1	**Y**
0	0	1	**0**
0	1	1	**1**
1	0	1	**1**
1	1	1	**0**

- *Error is determined by going through all the 4 samples*
- Here is a model 1 with weight values

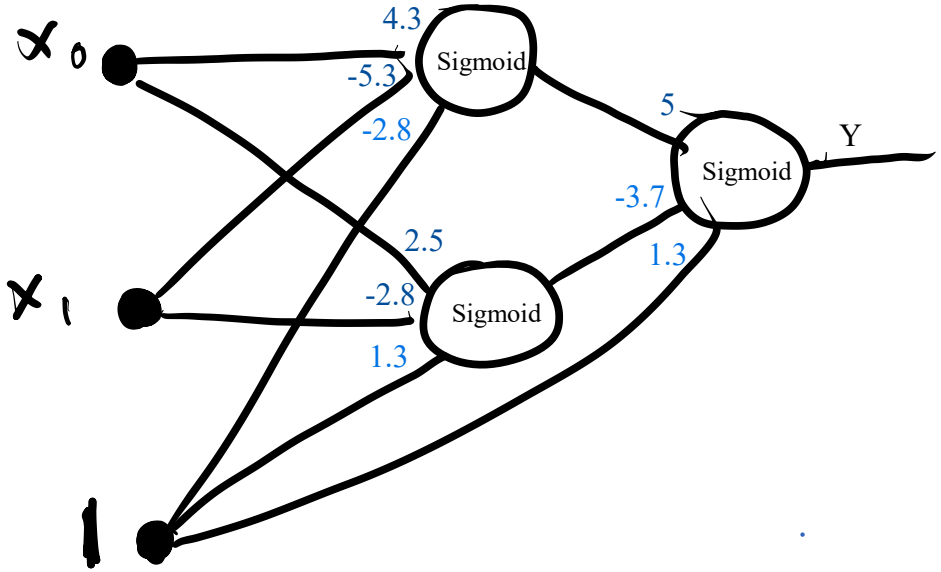


Self-Exercise 2: Write a simple Python code to calculate MSE, RMSE, MAE, and MAPE

- Input data examples and expected outputs:

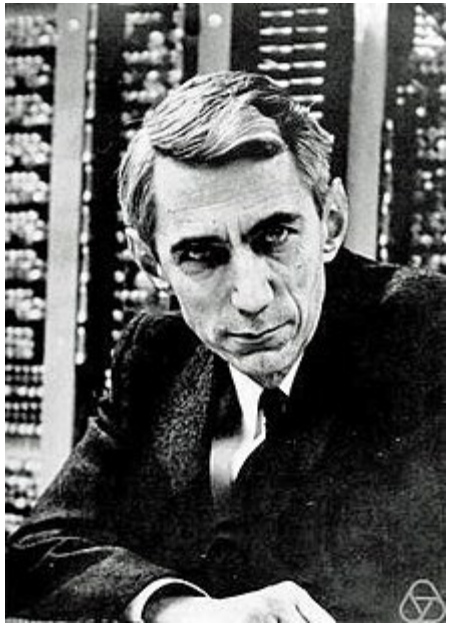
X_0	X_1	1	**Y**
0	0	1	**0**
0	1	1	**1**
1	0	1	**1**
1	1	1	**0**

- *Error is determined by going through all the 4 samples*
- Here is a model 1 with weight values



Information Theory by mathematician Claude Shannon

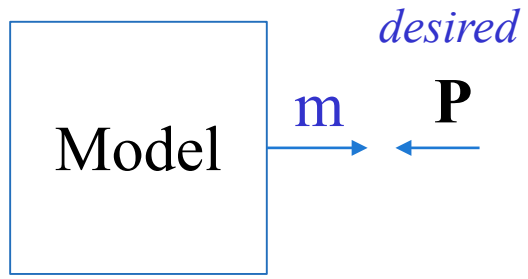
- Information **entropy** is a concept from information theory - It tells how much information there is *in an event*.
- In general,
  - the more *uncertain or random* the event is, the more information it will contain. (Entropy or uncertainty is increased)
  - The event is more clearly stated, information is a decrease in *uncertainty or entropy*.
- https://youtu.be/2s3aJfRr9gE

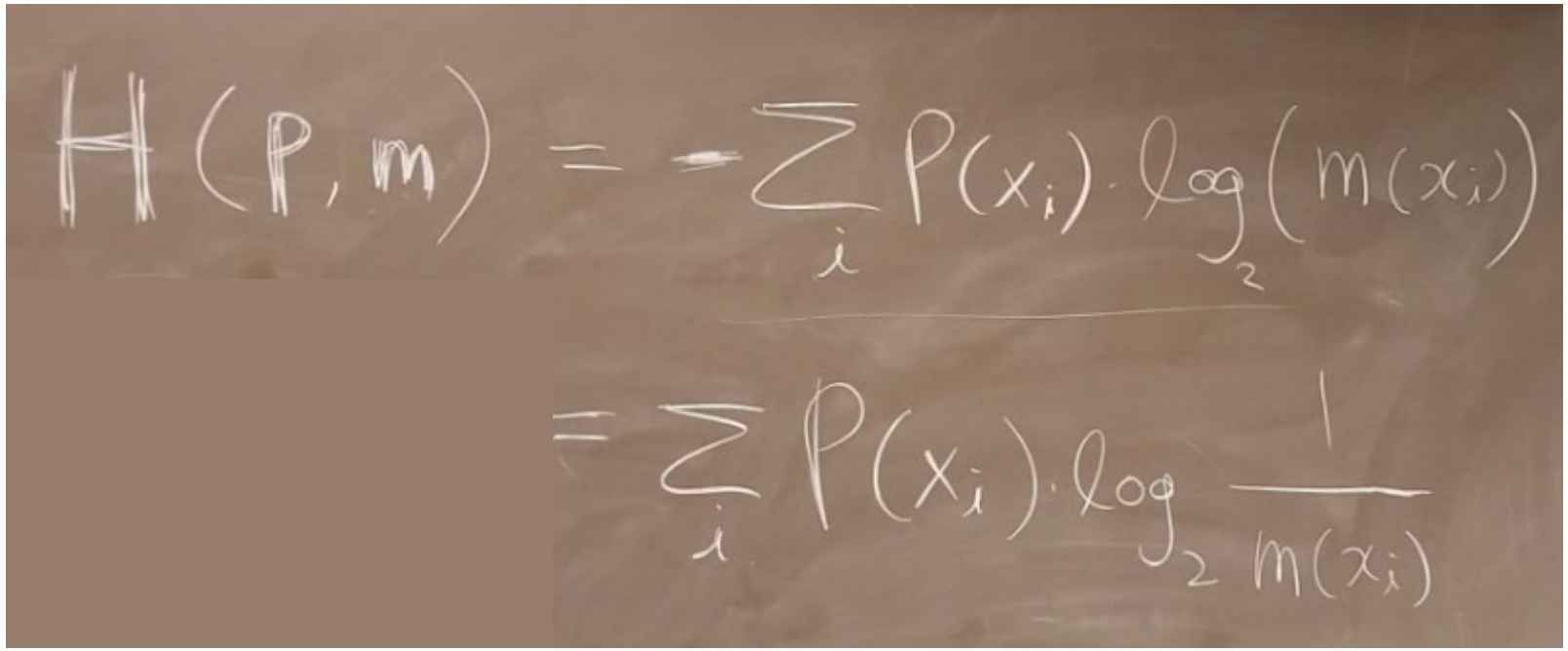


Born in Petoskey, Michigan

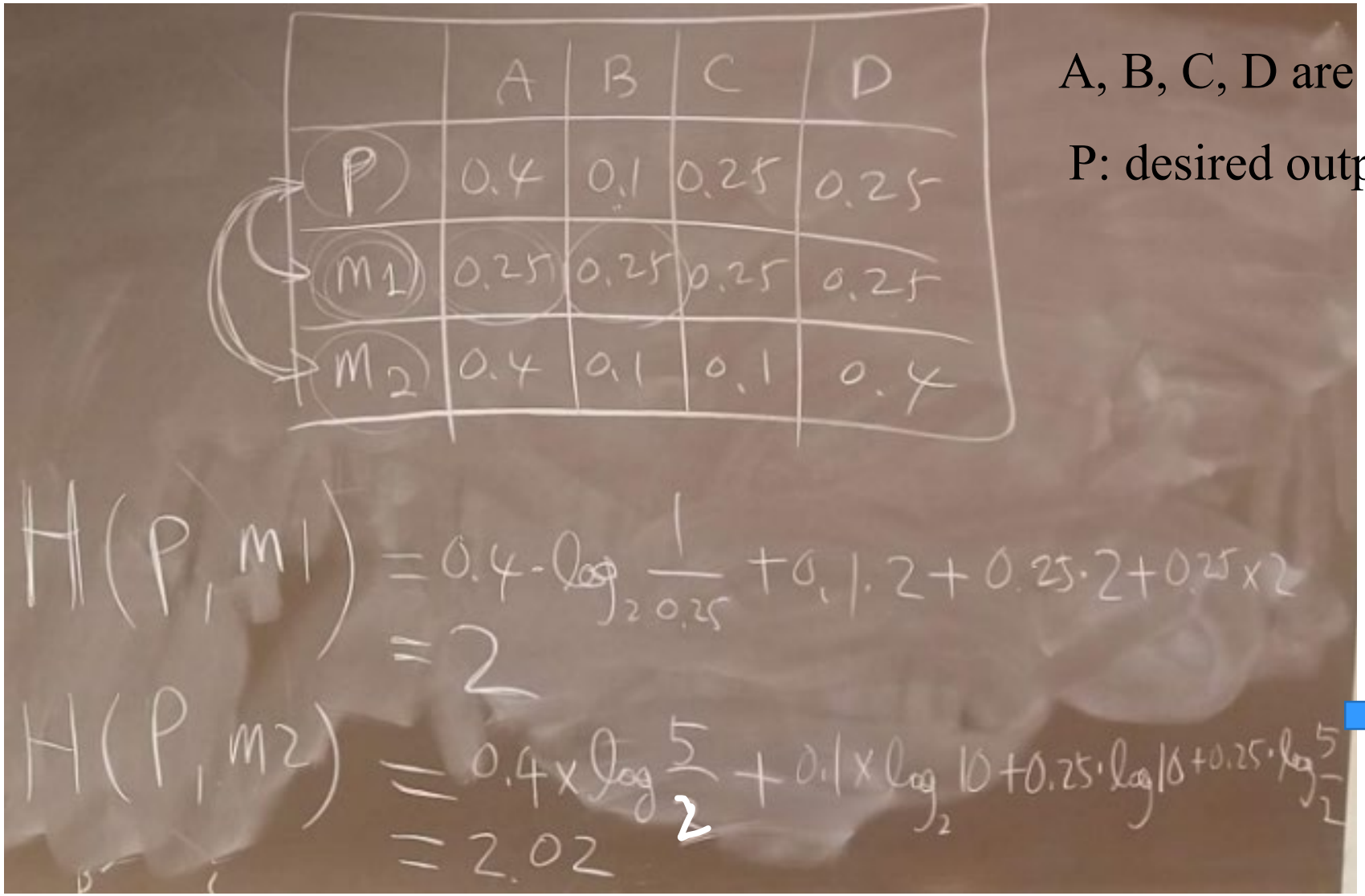
Cross Entropy

building upon entropy and generally calculating the difference between two probability distributions, **P** and m





Example: Cross Entropy



A, B, C, D are outputs

P: desired output

```
mse(P, m1): 0.01
mse(P, m2): 0.01
```

```
mae(P, m1): 0.08
mae(P, m2): 0.08
```

→ m1 is better

MSE doesn't punish misclassifications enough

Why Cross-entropy rather than mse?

- "Sigmoid + mse" will converge slower compare to "sigmoid + cross_entropy" due to the *gradient vanishing* issue
- MSE doesn't punish misclassifications enough
- the cross-entropy arises as the natural cost function to use if you have a sigmoid or softmax nonlinearity in the output layer of your network, and you want to maximize the likelihood of classifying the input data correctly

- https://susanqq.github.io/tmp_post/2017-09-05-crossentropyvsmes

Review: SSE, MSE, and MAE

```
y_desired = np.array([3,   4,   5,   6  ])
y =         np.array([5,   4,   5.1, 5.5])

def sumSqErr(y_d, y):
  sum_sq_err = 0.0
  for i in range (len(y_d)):
    sum_sq_err += (y_d[i]-y[i])**2
  return sum_sq_err

def meanSqErr(y_d, y):
  return sumSqErr(y_d, y) / len(y_d)
```

```
sse = ((y - y_desired)**2).sum()
mse = ((y - y_desired)**2).mean()
```

Please test **SSE_MSE_MAE.ipynb**

```
def sumSqErr(y_t, y_p):
  sum_sq_err = 0.0
  for i in range (len(y_t)):
    sum_sq_err += (y_t[i]-y_p[i])**2
  return sum_sq_err

def mse(y_t, y_p):
  return sumSqErr(y_t, y_p) / len(y_t)

# calculate cross entropy
def cross_entropy(p, q):
  return -sum([p[i]*np.log2(q[i]) for i in range(len(p))])
```

```
P = np.array([0.4,   0.1,   0.25,   0.25]) # y_true
m1 = np.array([0.25,   0.25,   0.25, 0.25]) # y_predict
m2 = np.array([0.4,   0.1,   0.1, 0.4]) # y_predict
```

```
print("mse(P, m1): %.2f" %mse(P, m1))
print("mse(P, m2): %.2f" %mse(P, m2))

print("mae(P, m1): %.2f" % np.absolute(P-m1).mean())
print("mae(P, m2): %.2f" % np.absolute(P-m2).mean())

print("crossEntropy(P, m1): %.2f" % cross_entropy(P, m1))
print("crossEntropy(P, m2): %.2f" % cross_entropy(P, m2))
```

```
mse(P, m1): 0.01
mse(P, m2): 0.01
mae(P, m1): 0.08
mae(P, m2): 0.08
crossEntropy(P, m1): 2.00
crossEntropy(P, m2): 2.02
```

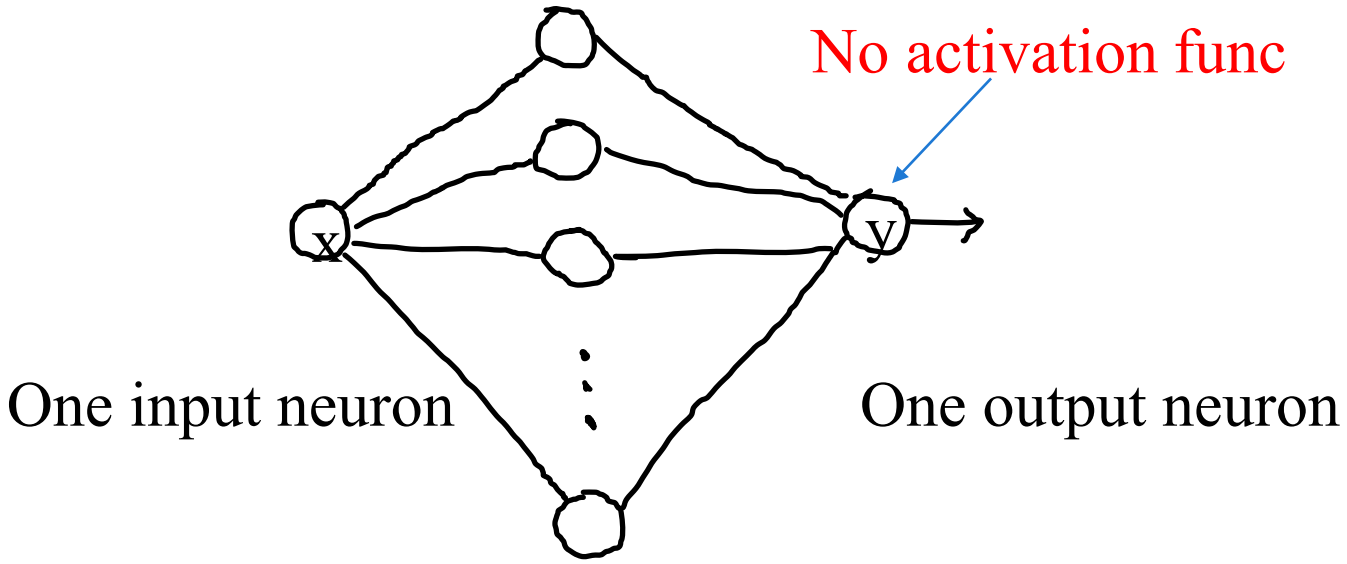
Review:

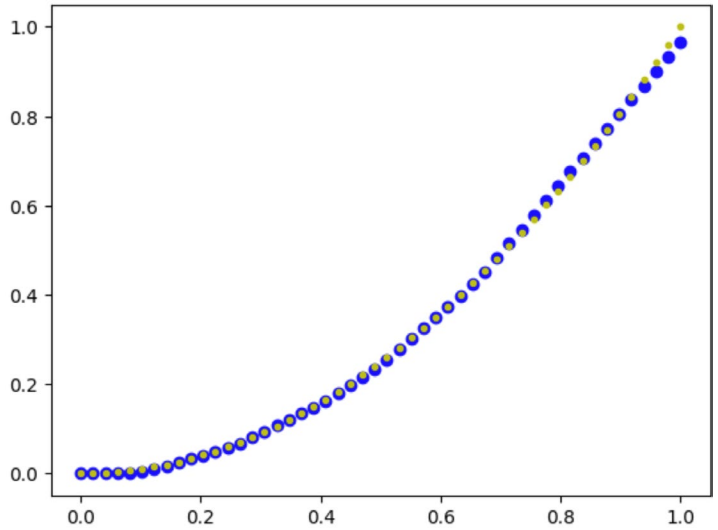
Table 4.1 Choosing the right last-layer activation and loss function for your model

Problem type	Last-layer activation	Loss function
Binary classification	`sigmoid`	`binary_crossentropy`
Multiclass, single-label classification	`softmax`	`categorical_crossentropy`
Multiclass, multilabel classification	`sigmoid`	`binary_crossentropy`
Regression to arbitrary values	None	`mse` or `mae`
Regression to values between 0 and 1	`sigmoid`	`mse` or `binary_crossentropy`

Like linear function

Regression to arbitrary values, for example: y = x^2






```
m3 = Sequential([
    Dense(32, input_dim=1, activation='relu'),
    Dense(1)
])
m3.compile(optimizer=optimizers.SGD(learning_rate=0.1), loss='mse')
m3.fit(X, Y3, epochs=500, batch_size=5, verbose = 0)
```

Review – "sparse_categorical_crossentrophy"

Table 4.1 Choosing the right last-layer activation and loss function for your model

Problem type	Last-layer activation	Loss function
Binary classification	`sigmoid`	`binary_crossentropy`
Multiclass, single-label classification	`softmax`	`categorical_crossentropy`
Multiclass, multilabel classification	`sigmoid`	`binary_crossentropy`
Regression to arbitrary values	None	`mse` or `mae`
Regression to values between 0 and 1	`sigmoid`	`mse` or `binary_crossentropy`