# Machine Learning Using Tensorflow

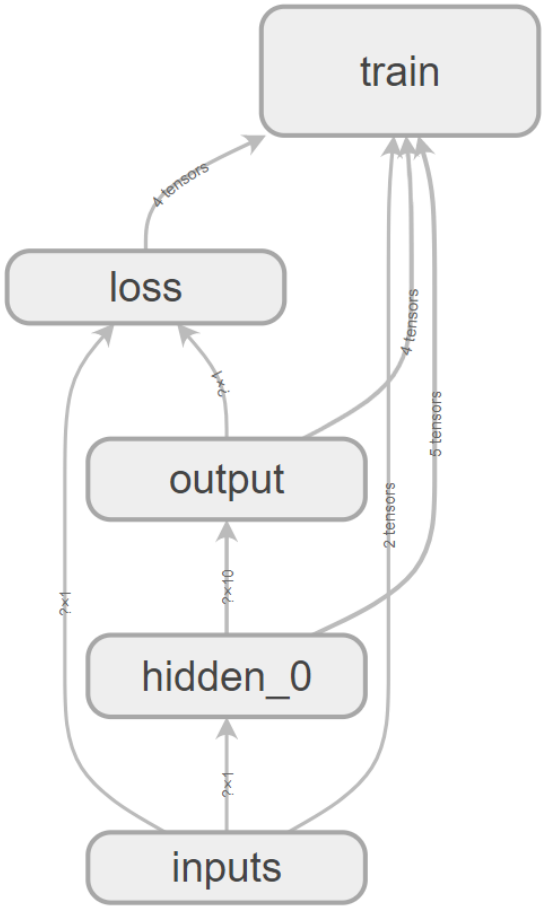
## Week3:
## Basis of  Deep Learning & classification

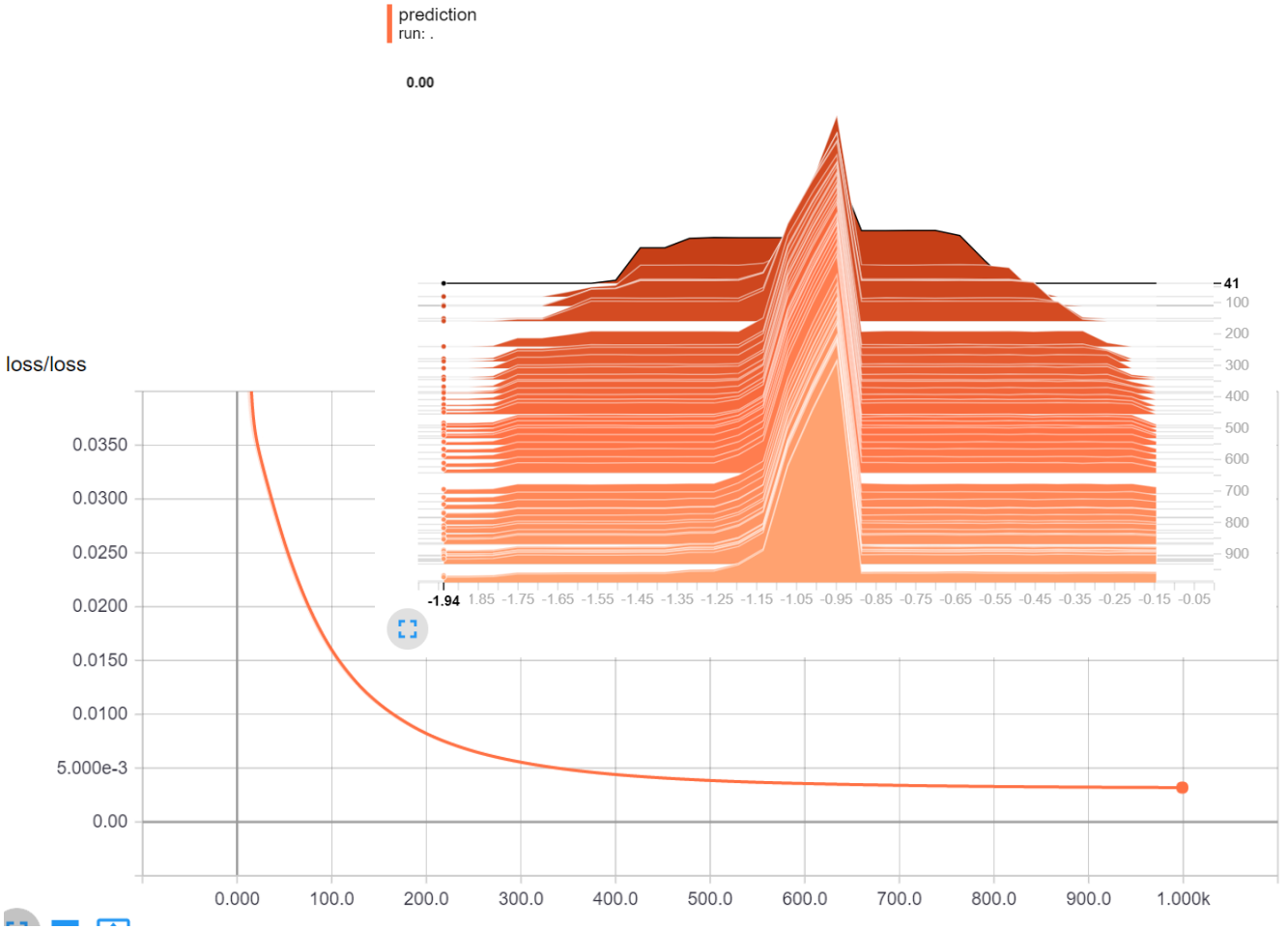
**Shu-Ting Pi, PhD**
**UC Davis**

TensorFlow™

# Tensorboard

# Visualization using Tensorboard



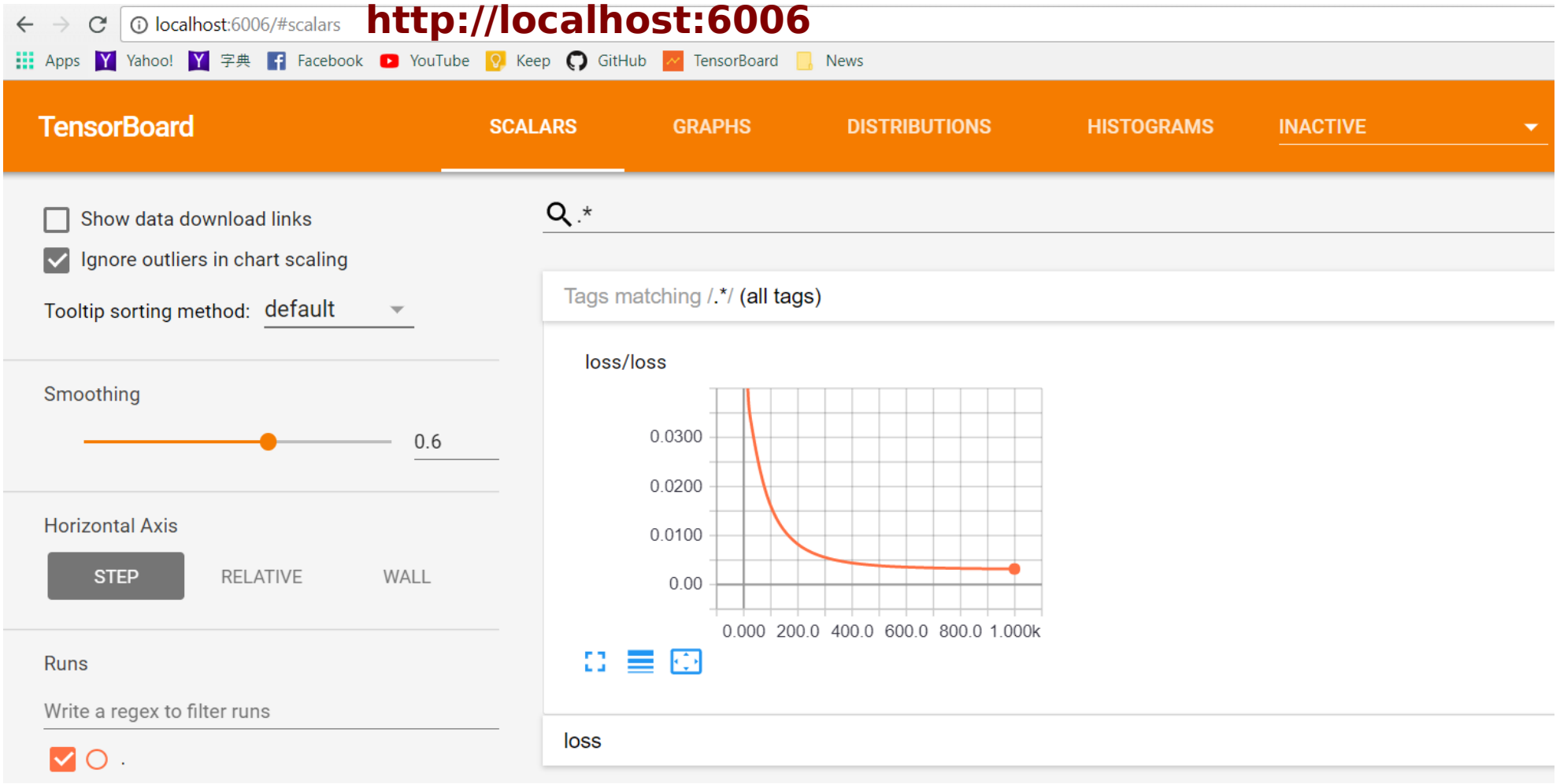



# Visualization using Tensorboard

```
39  # computation graph ==================
40  # define placeholder for inputs to network
41  with tf.name_scope('inputs'):
42      x_tf = tf.placeholder(tf.float32, [None, 1])
43      y_tf = tf.placeholder(tf.float32, [None, 1])
44
45  # neural network layers
46  l1 = tf.layers.dense(x_tf, 10, act_func,name='hidden_0')          # hidden layer
47  prediction = tf.layers.dense(l1, 1, name='output')                    # output layer
48
49  # the error between prediction and real data
50  with tf.name_scope('loss'):
51      loss = tf.reduce_mean(tf.reduce_sum(tf.square(y_tf-prediction), reduction_indices=[1]))
52
53  with tf.name_scope('train'):
54      train_op = tf.train.GradientDescentOptimizer(learning_rate).minimize(loss)
55
56
57  # tensorflow session =================
58  # important step
59  sess = tf.Session()
60
61  # generate summary
62  summary_write=tf.summary.FileWriter('log/',sess.graph)
```

# Visualization using Tensorboard

```
C:\Users\pipidog\Dropbox\Code\MLclass\codes>tensorboard --logdir logs
```



# categorize

# How to categorize data?

**Example:**

Input (X) : hosing price, years,...etc.

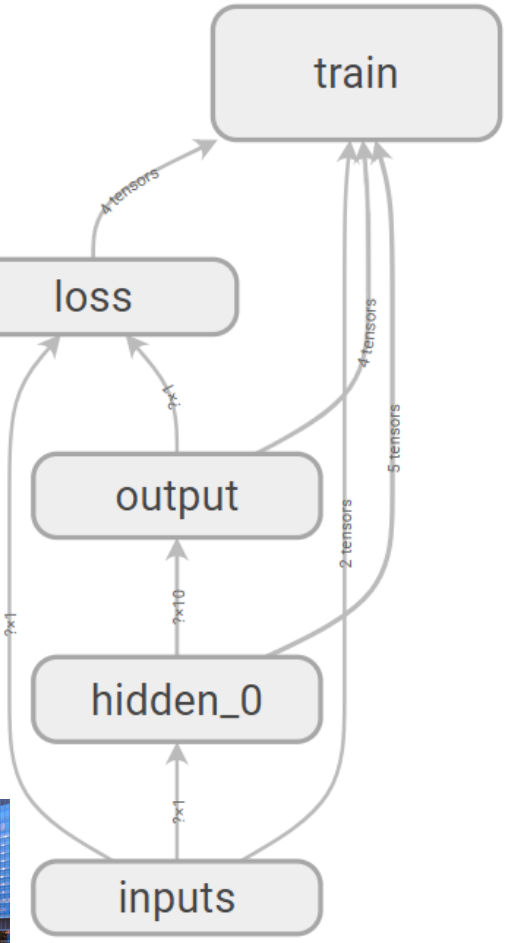
Target (Y): identify the city of the house's loc

**Question:**

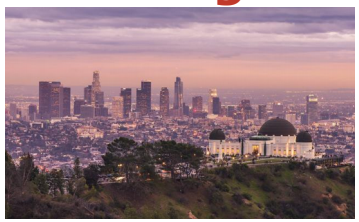
How to "output" a label ?

How to define the "loss" bwtween two label

How to "optimize" the parameters to genera




**Los Angels**



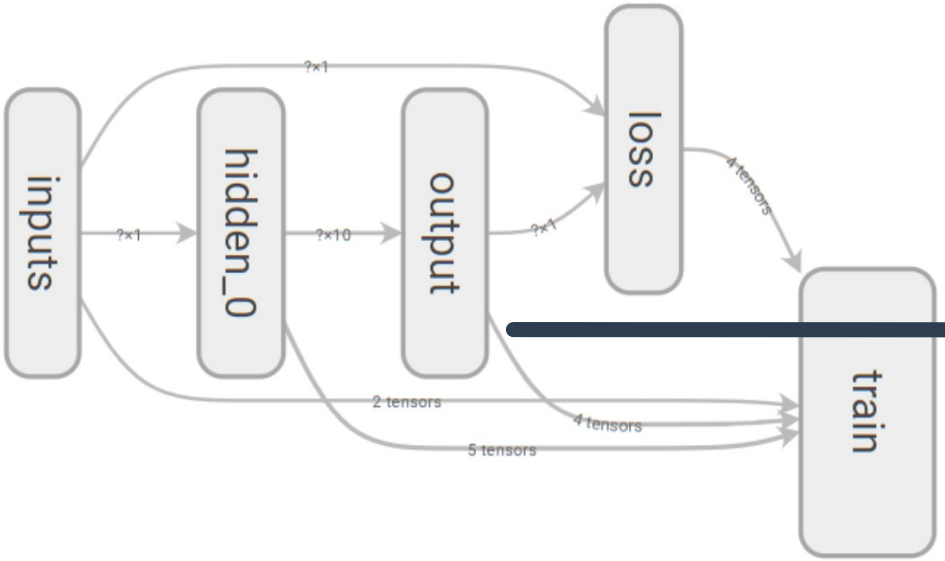
**San Francisco**

**New York**

**Chicago**

# Binary representation

**“Probabilize” your data !**




New York $\rightarrow \begin{pmatrix} 1 & 0 & 0 & 0 \end{pmatrix}$

Los Angel $\rightarrow \begin{pmatrix} 0 & 1 & 0 & 0 \end{pmatrix}$

San Francisco $\begin{pmatrix} 0 & 0 & 1 & 0 \end{pmatrix}$

Chicago $\rightarrow \begin{pmatrix} 0 & 0 & 0 & 1 \end{pmatrix}$

$$\begin{pmatrix} 0.12 & 0.08 & \boxed{0.67} & 0.13 \end{pmatrix}$$

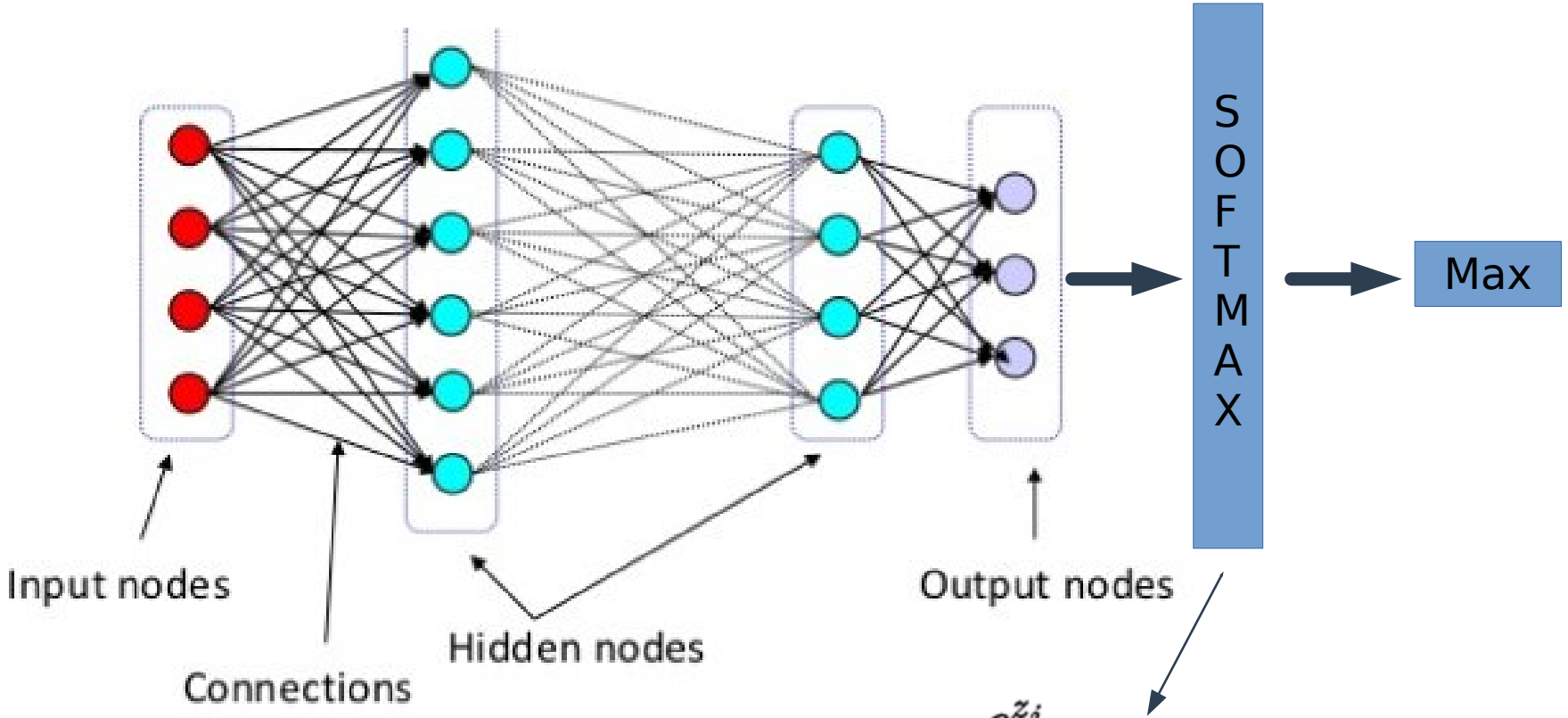
**San Francisco !**

**Question:**

**1. How to “Probabilize” your output? (softmax !)**

**2. How to measure the difference between two probability distribution? (**

# Softmax function

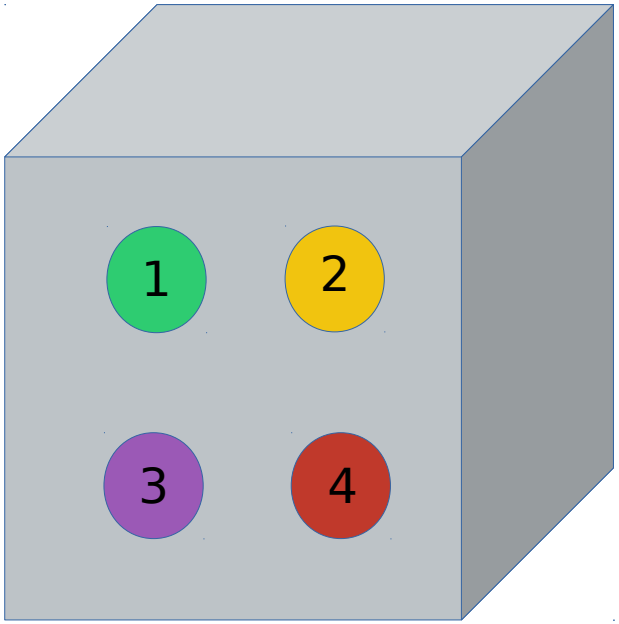



$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, \ldots, K.$$

# Shannon Game

Let's play a game:

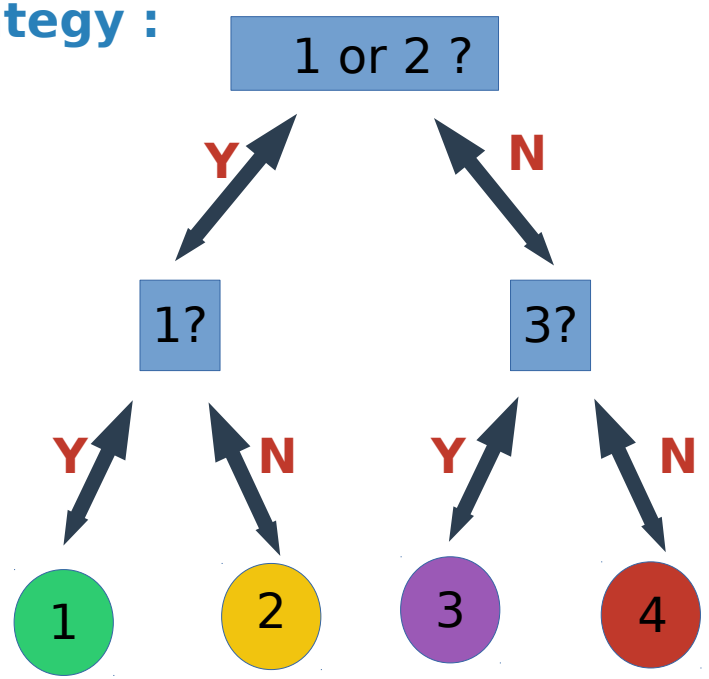
- There are many balls with four different colors in a box.
- I will pick one from the box.
- You ask a question and I will tell you “yes” or “no” (binary)




# Shannon Game

**Know Information:**
No information about the population of colors

**Best Strategy :**

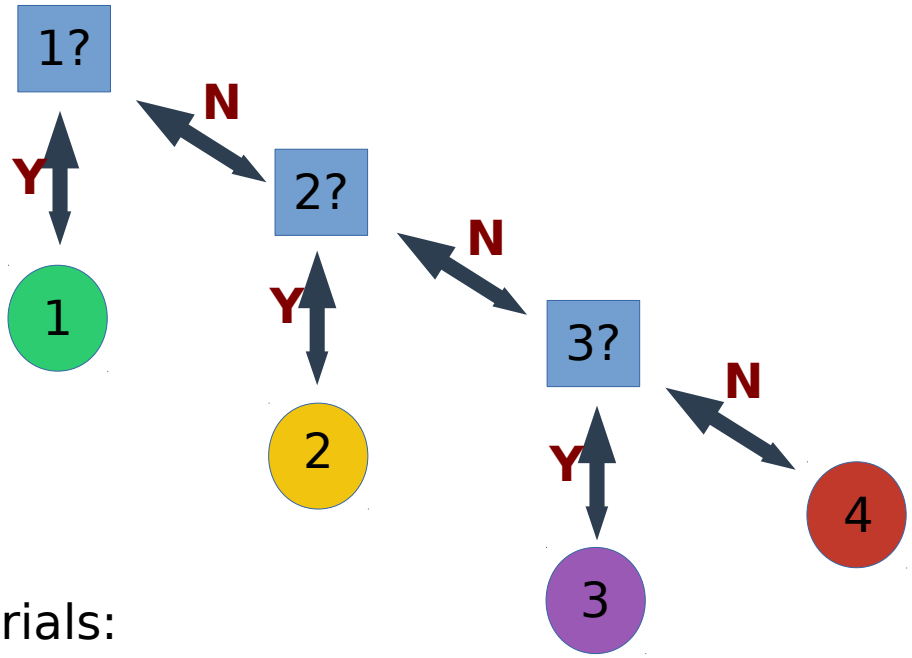



Average trials: $\frac{1}{4} \times 2 + \frac{1}{4} \times 2 + \frac{1}{4} \times 2 + \frac{1}{4} \times 2 = 2$

# Shannon Game

**Know Information:**
Green: 1/2, Yellow: 1/4, Purple: 1/8, Red: 1/8

**Best Strategy :**




Average trials:

$$\frac{1}{2} \times 1 + \frac{1}{4} \times 2 + \frac{1}{8} \times 3 + \frac{1}{8} \times 3 = 1.75 < 2$$

# Shannon Game

**Know Information:**
All balls are red

**Best Strategy :**

**No guess needed ! Average Trial=0**

For probability p, we need to gue $log_2 \frac{1}{p}$

The average total trials:

$$p \times log_2 \frac{1}{2}$$

Information Entropy


Meaning:
1. A measurement of the “uncertainty”
2. Always positive or zero (zero = no uncertainty)
3. The penalty if you want to eliminate uncertainty

# Shannon Entropy

**For probability p, you need to gu**

$$log_2 \frac{1}{p}$$

**Average trials:**

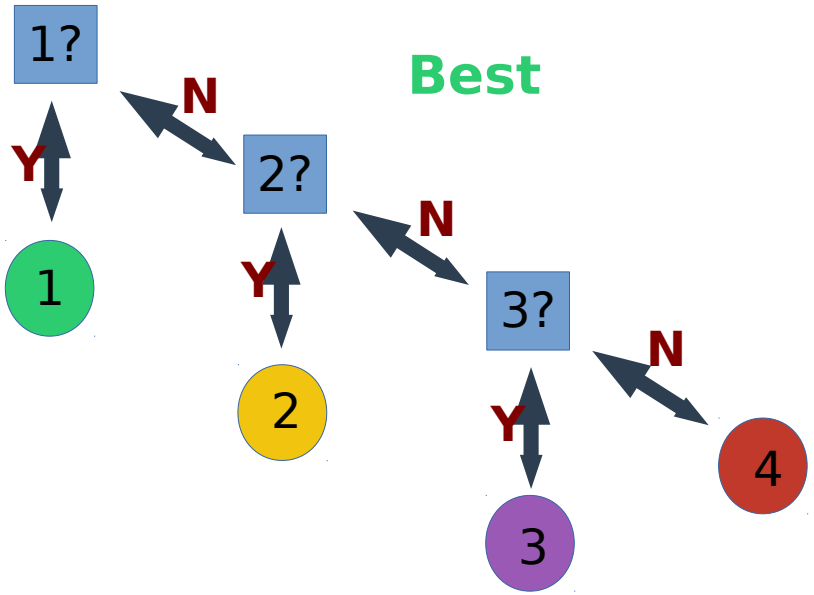
$$p \times log_2 \frac{1}{2}$$

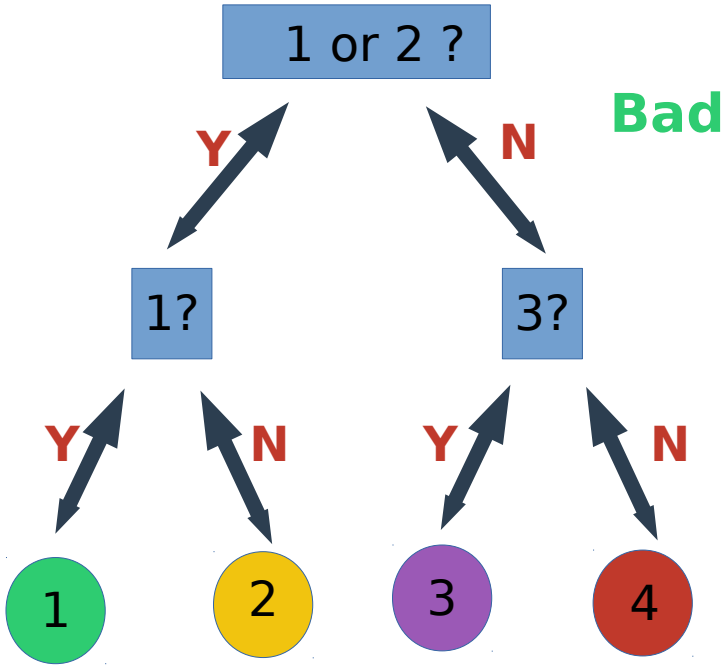
**Information Entropy**

Meaning:
1. A measurement of the “uncertainty”
2. Always positive or zero (zero = no uncertainty)
3. The penalty if you want to eliminate uncertainty

# What if we choose a bad strategy

**Know Information:**
Green: 1/2, Yellow: 1/4, Purple: 1/8, Red: 1/8




True probability is p, but we choose q (use q to represent p):

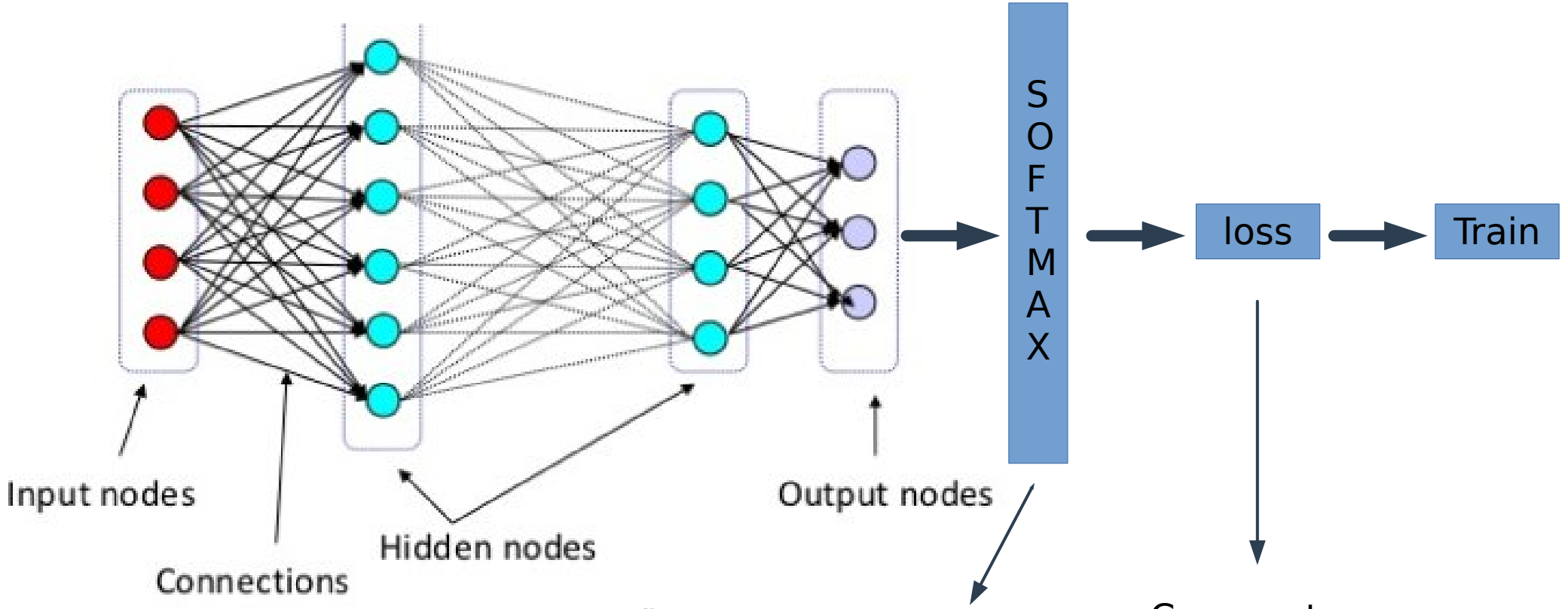
$$S(p,q) = \sum_{k=1}^{N} p_k log_2 q_k = 2 > 1.75$$

**Cross entropy**

$$KL(p||q) = S(p,q) - S(p) = 0.25$$

**Relative entropy , KL divergence**

# Tensorflow Graph




$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, \ldots, K.$$

Cross entropy

$$S(p,q) = \sum_{k=1}^{N} p_k log_2 q_k$$

# Exercise

```
21  import tensorflow as tf
22  import matplotlib.pyplot as plt
23  import numpy as np
24  import sys
25
26  # parameters ============================
27  # generate data
28  data_center=[[2,2],[2,-2],[-2,2],[-2,-2]] # data distribtion centers
29  train_num=1000                    # number of data around a data center
30  test_num=100
31  noise_std=0.7                   # noise of data around a center
32  # hidden layer
33  layer_nodes=[10]
34  act_func=tf.nn.relu
35  # train
36  batch_size=50
37  step=500
38  step_show=10
39  learning_rate=0.5
40  # generate data ============================
41  np.random.seed(1)
42  tot_class=len(data_center)
43
44  # create empty numpy array, so we can use vstack and hstack
45  x_train=np.array([]).reshape(0,2)
46  x_test=np.array([]).reshape(0,2)
47  y_train=np.array([])
48  y_test=np.array([])
49
50  # create training data and validation dataset around each data_center
51  for n, dc in enumerate(data_center):
52          x_train=np.vstack((x_train,np.random.normal(np.tile(dc,(train_num,1)),noise_std)))
53          y_train=np.hstack((y_train,np.repeat(n,train_num)))
54          x_test=np.vstack((x_test,np.random.normal(np.tile(dc,(test_num,1)),noise_std)))
55          y_test=np.hstack((y_test,np.repeat(n,test_num)))
```

# Exercise

```
57  # computation graph ========================
58  tf.set_random_seed(1)
59
60  x_tf = tf.placeholder(tf.float32, [None,2])      # input x
61  y_tf = tf.placeholder(tf.int32, None)            # input y
62
63  # hidden layers
64  h=[]
65  inp_dat=x_tf
66  for nodes in layer_nodes:
67      h.append(tf.layers.dense(inp_dat,nodes,act_func))
68      inp_dat=h[-1]
69
70  # output
71  output = tf.layers.dense(h[-1],tot_class)
72  # loss
73  loss = tf.losses.sparse_softmax_cross_entropy(labels=y_tf, logits=output)
74  # predictions
75  prediction = tf.argmax(output, axis=1)
76  # return (acc, update_op), and create 2 local variables
77  accuracy = tf.metrics.accuracy(labels=tf.squeeze(y_tf), predictions=prediction)[1]
78  # train operator
79  train_op = tf.train.AdamOptimizer(learning_rate=learning_rate).minimize(loss)
```

# Exercise

```
81  # TF session ================================
82  sess = tf.Session()
83  init_op = tf.group(tf.global_variables_initializer(), tf.local_variables_initializer())
84  sess.run(init_op)
85
86  plt.ion()
87  acc=np.array([]).reshape(0,2)
88  plt.figure(0,figsize=(18, 6))
89  plt.subplot(1,3,1)
90  plt.scatter(x_test[:, 0], x_test[:, 1], c=y_test, s=100, lw=0, cmap='tab10')
91  plt.title('true test dataset',fontdict={'size': 14, 'color': 'green'})
92  for n in range(step+1):
93      batch_select=np.random.randint(0,len(x_train)-1,batch_size)
94
95      # train and net output
96      _, acc_train = sess.run([train_op, accuracy], {x_tf: x_train[batch_select], y_tf: y_train[batch_select]})
97
98      if n % step_show == 0:
99          # plot and show learning process
100         _, acc_test, pred = sess.run([train_op, accuracy, output], {x_tf: x_test, y_tf: y_test})
101         acc=np.vstack((acc,np.array([acc_train,acc_test])))
102
103         plt.subplot(1,3,2)
104         plt.cla()
105         plt.scatter(x_test[:, 0], x_test[:, 1], c=pred.argmax(1), s=100, lw=0, cmap='tab10')
106         plt.title('step={0:4d}, accuracy={1:.3f}'.format(n, acc_test),fontdict={'size': 14, 'color': 'green'})
107         plt.pause(0.1)
```