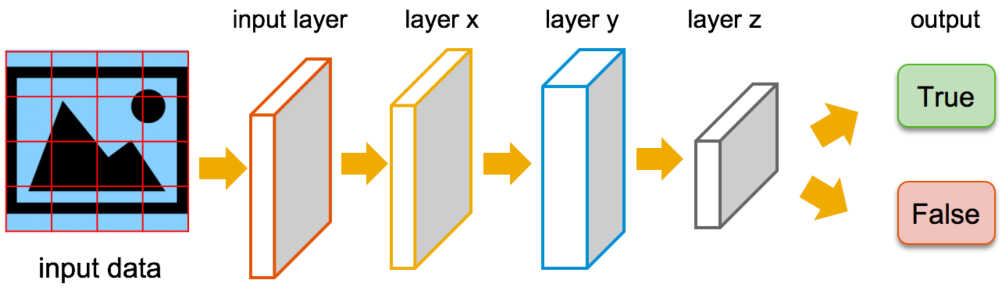
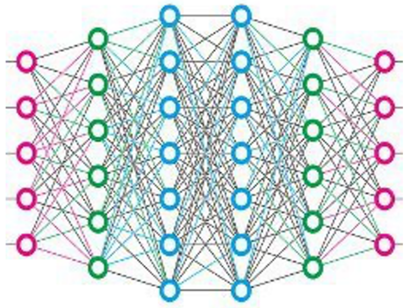
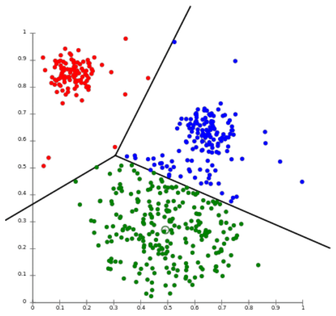
CS 445
Introduction to Machine Learning

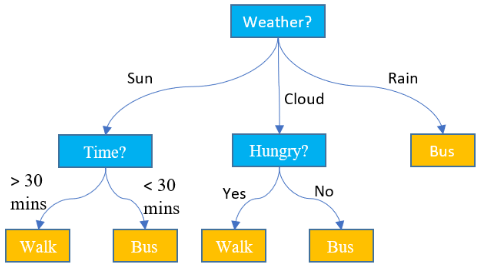


Softmax and
One-Hot Encoding
Convolution Neural Networks

Instructor: Dr. Kevin Molloy







Announcements

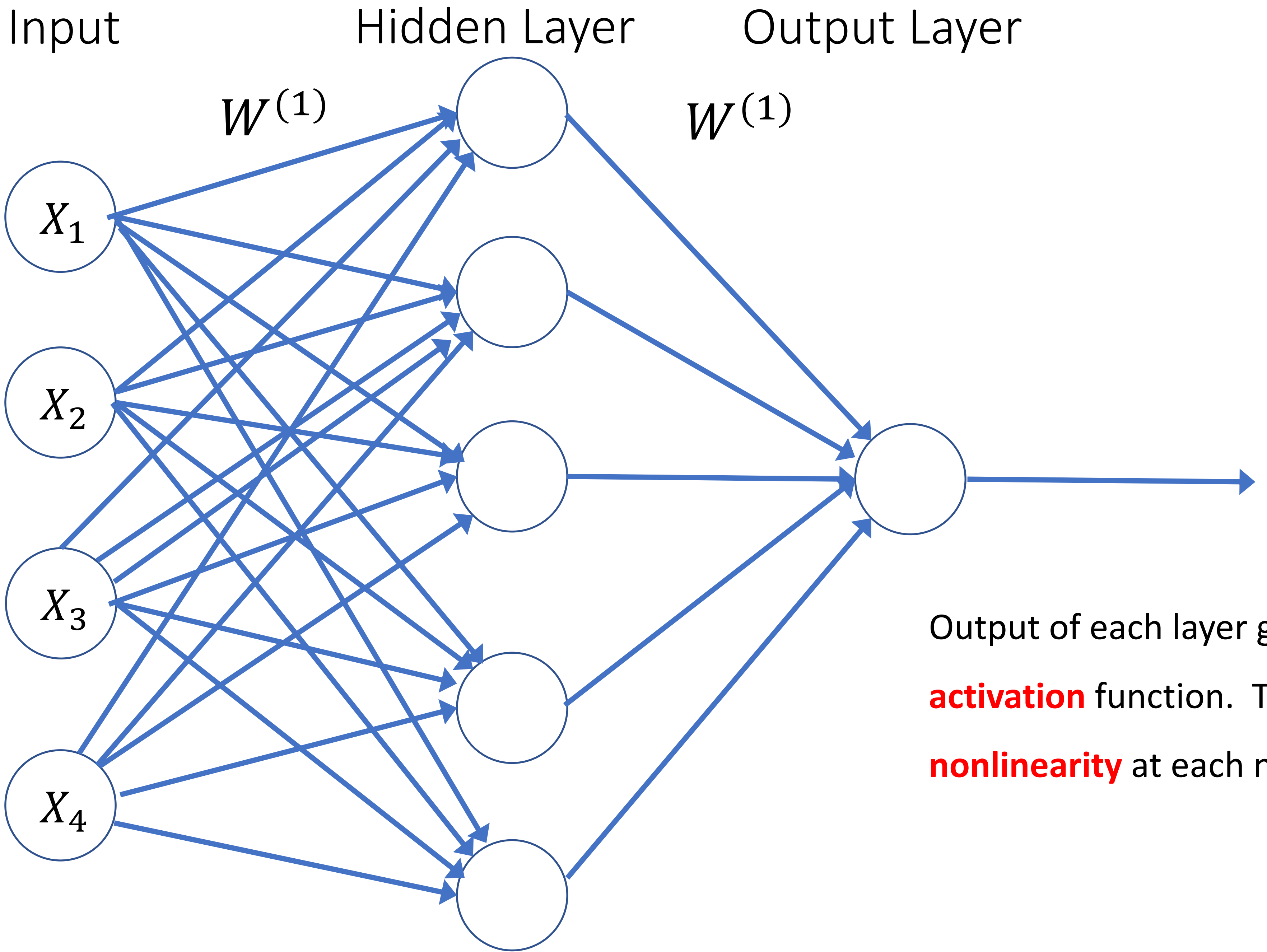
PA 3 Posted

- Due a week from Friday at 5:00 pm

Learning Objectives

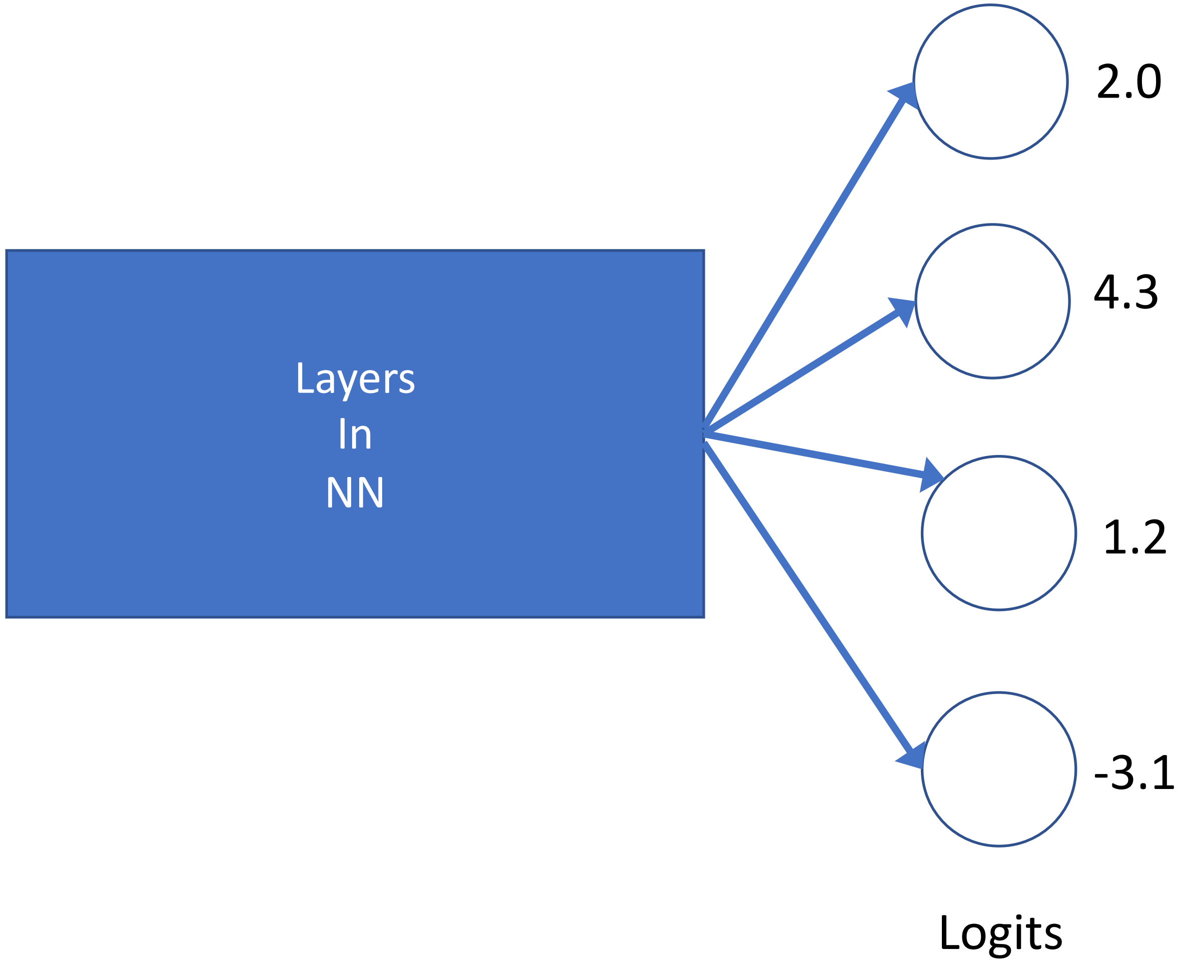
- Multiclass classification with NNs
- One hot encoding
- Monitoring Keras
- Sequential Networks and Image Recognition
- Utilize Convolution in NN (called CNNs)

Binary Classification



Output of each layer goes through an **activation** function. This introduces a **nonlinearity** at each node.

Multiclass Classification



Logit – one per class. Each value tells us something about the target class. We would like to change these values to a probability distribution.

$$Softmax(a_i) \frac{\exp(a_i)}{\sum_j \exp(a_j)}$$

Dealing with Categorical Data

- Problems like image (or digit) recognition have multiple categorical labels.
- Neural networks require all data (including the labels to be numeric).

How to Convert?   In general, two steps are required

1. Convert labels to an integer encoding
2. One-hot encoding

Integer and One-hot encoding

Classifying dogs, cats, and hamsters.

- "Dog" can be 1
- "Cat" can be 2
- Hamster can be "3".

Is this enough?

Integer and One-hot encoding

Classifying dogs, cats, and hamsters.

- "Dog" can be 1
- "Cat" can be 2
- Hamster can be "3".

Is this enough? Turns out no. This encoding includes an ordinal relationship, which is not really applicable. One-hot encoding is the answer.

- Create a binary variable for each class, all variables are set to zero except for the actual class, which is set to 1.

Keras Examples

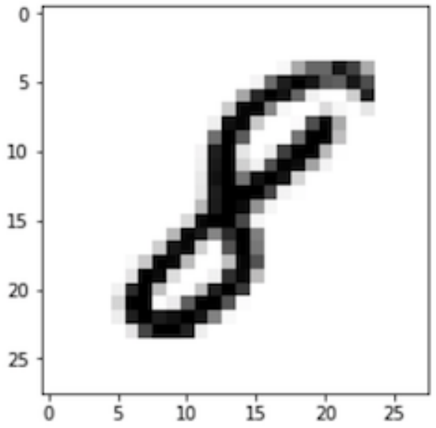
```
y_train = np_utils.to_categorical(y_train, 3)# 3 class problem

model.add(Dense(3, activation='softmax')) # add softmax layer as last layer

tensorboard = TensorBoard(log_dir='./logs', histogram_freq=1,
  write_images=True)

history = model.fit(X_train, y_train, epochs=10000, batch_size=1000,
verbose=1, callbacks=[tensorboard])
```

Images



28 x 28 x 1 = 784

3024 x 4032 x 3 = 36,578,304

A fully connected network with 1,000 hidden nodes in the first layer, W is a 36 million x 1000 (36 billion entries)

Image Convolutions

Picture convolved with a "Canny edge detector".

How can we do this?

Detecting Vertical Edges

3	0	1	2
1	5	8	9
2	7	2	5
0	1	3	1

Gray-scaled image

*

1	0	-1
1	0	-1
1	0	-1

Filter or Kernel

=

Detecting Vertical Edges

3	0	1	2
1	5	8	9
2	7	2	5
0	1	3	1

Gray-scaled image

*

1	0	-1
1	0	-1
1	0	-1

Filter or Kernel

=

-5	

$$3*1 + 1*1 + 2*1 + 0*0 + 5*0 + 7*0 + 1*-1 + 8*-1 + 2*-1 = -5$$

Detecting Vertical Edges

3	0	1	2
1	5	8	9
2	7	2	5
0	1	3	1

Gray-scaled image

*

1	0	-1
1	0	-1
1	0	-1

Filter or Kernel

=

-5	-4

0*1 + 5*1 + 7*1 + 1*0 + 8*0 + 2*0 + 2*-1 + 9*-1 + 5*-1 = -4

Detecting Vertical Edges

3	0	1	2
1	5	8	9
2	7	2	5
0	1	3	1

Gray-scaled image

*

1	0	-1
1	0	-1
1	0	-1

Filter or Kernel

=

-5	-4
-10	

1*1 + 2*1 + 0*1 + 5*0 + 7*0 + 1*0 + 8*-1 + 2*-1 + 3*-1 = -10

Detecting Vertical Edges

3	0	1	2
1	5	8	9
2	7	2	5
0	1	3	1

Gray-scaled image

*

1	0	-1
1	0	-1
1	0	-1

Filter or Kernel

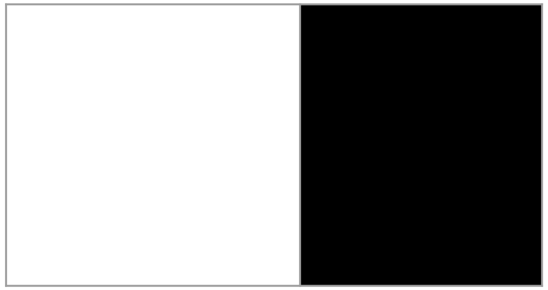
=

-5	-4
-10	-2

5*1 + 7*1 + 1*1 + 8*0 + 2*0 + 3*0 + 1*-1 + 3*-1 + 1*-1 = -2

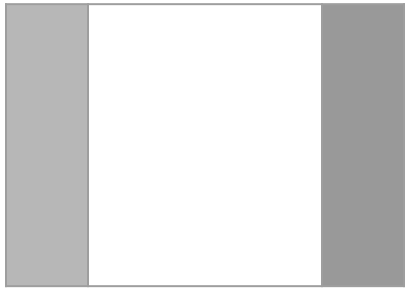
Detecting Vertical Edges

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0



1	0	-1
1	0	-1
1	0	-1

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0



Slides adapted from Andrew Ng

Other Filters

1	0	-1
1	0	-1
1	0	-1

Vertical Edges

1	1	1
0	0	0
-1	-1	-1

Horizontal Edges

1	2	1
0	0	0
-1	-2	-1

Sobel Filter

Novel Idea

1	0	-1
1	0	-1
1	0	-1

Vertical Edges

1	1	1
0	0	0
-1	-1	-1

Horizontal Edges

1	1	1
0	0	0
-1	-1	-1

Sobel Filter

How about **learning** a set of filters for edge/object characteristics?

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

Losing Some Data Along the Way

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

6 x 6 (n x n)

1	0	-1
1	0	-1
1	0	-1

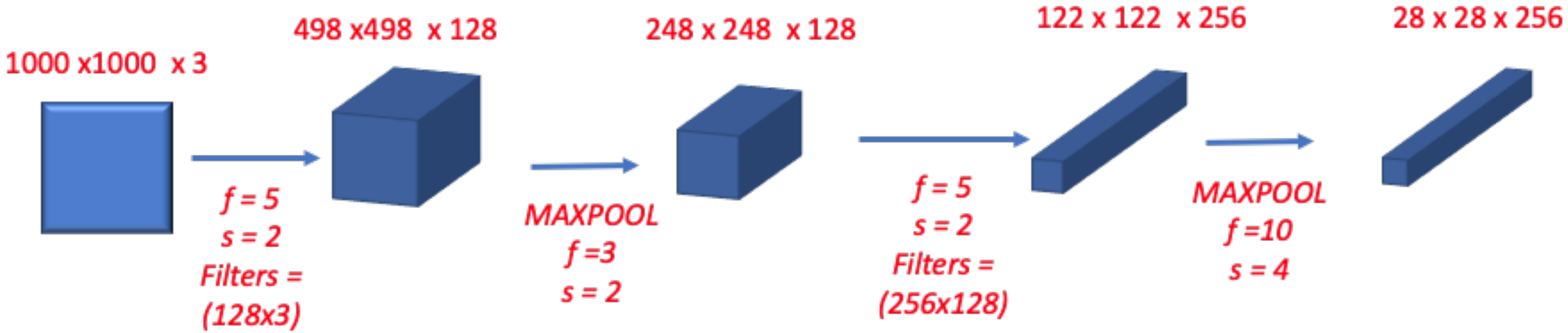
3 x 3 (f x f)

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0

4 x 4

In general
(n – f + 1, n – f +1)

- Notice corners only appear in 1 convolution computation.
- If you piece together several layers, you keep consolidating the signal/information


1000 x1000 x 3
498 x498 x 128
248 x 248 x 128
122 x 122 x 256
28 x 28 x 256
f = 5
s = 2
Filters =
(128x3)
MAXPOOL
f =3
s = 2
f = 5
s = 2
Filters =
(256x128)
MAXPOOL
f =10
s = 4