MACHINE LEARNING & DATA MINING
CS/CNS/EE 155

Deep Learning
Part II
**recap of last lecture**

Logistic regression can’t handle non-linear data distributions.

<table>
<thead>
<tr>
<th></th>
<th>AND</th>
<th>OR</th>
<th>XOR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><img src="#" alt="AND diagram" /></td>
<td><img src="#" alt="OR diagram" /></td>
<td><img src="#" alt="XOR diagram" /></td>
</tr>
<tr>
<td></td>
<td>linearly separable</td>
<td>linearly separable</td>
<td>not linearly separable</td>
</tr>
</tbody>
</table>
recap of last lecture

let’s use non-linear features to linearize the problem!

one approach: use a set of **hand-crafted** non-linear transformations

\[ X_1, X_2 \rightarrow X_1, X_2, X_1X_2 \]

linear decision boundary \hspace{1cm} hyperbolic decision boundary

another approach: use a set of **learned** non-linear transformations

\[ X_1, X_2 \rightarrow X_1 \land X_2, X_1 \lor X_2 \]

linear decision boundary \hspace{1cm} multiple linear decision boundaries

\[ \neg(X_1 \land X_2) \land (X_1 \lor X_2) \]
recap of last lecture

'neuron'

weights

summation

non-linearity

output feature

input features

$X_1$

$X_2$

$X_3$

$X_4$

$\ldots$

$X_p$
recap of last lecture

‘neural network’

\[ \sum \]

depth 5
neural networks are function approximators that can be trained to match the data’s label distribution

\[ f(\text{data}) \sim P(\text{label} \mid \text{data}) \]

more parameters, depth \quad \rightarrow \quad more expressive, better approximation

(as long as you don’t overfit)
when is this useful?
Q: who is in this picture?

A: Yisong

Q: why? / how do you know?

A: umm...
**environment**

- Yisong
- pose, facial expression
- jacket, collared shirt, jeans, watch
- red chair, wood wall
- lighting

**observation**

- laws of nature
- black box

**inference**

- Yisong
- pose, facial expression
- jacket, collared shirt, jeans, watch
- red chair, wood wall
- lighting
- your brain
- black box
environment

Yisong
pose, facial expression
jacket, collared shirt, jeans, watch
red chair, wood wall
lighting

observation

laws of nature

inference

Yisong
pose, facial expression
jacket, collared shirt, jeans, watch
red chair, wood wall
lighting

‘who is this?’
the mappings between properties and images are too complicated to define manually

**deep learning to the rescue!**

two sides of the same coin

**generation**
there are latent properties that result in specific patterns in images

**discrimination**
there are patterns in images that allow us to infer latent properties
task:
train a deep neural network to discriminate whether or not an image contains Yisong
data

Yisong

not Yisong

labels
network architecture?
decide on an input size

**smaller input:**
- fewer parameters
- noisier patterns

**larger input:**
- more parameters
- clearer patterns

<table>
<thead>
<tr>
<th>15 x 15 x 3</th>
<th>50 x 50 x 3</th>
<th>75 x 75 x 3</th>
<th>100 x 100 x 3</th>
<th>150 x 150 x 3</th>
<th>205 x 205 x 3</th>
<th>280 x 280 x 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 x 3 x 3</td>
<td>3 x 3 x 3</td>
<td>3 x 3 x 3</td>
<td>3 x 3 x 3</td>
<td>3 x 3 x 3</td>
<td>3 x 3 x 3</td>
<td>3 x 3 x 3</td>
</tr>
</tbody>
</table>

- 15 25 35
- 15 25 35
- 3 3 3
- 1,875
- 3,675
- 7,500
- 16,875
- 30,000
- 67,500
- 126,075
- 235,200
decide on an input size

**larger input:**
- more parameters
- clearer patterns

**smaller input:**
- fewer parameters
- noisier patterns

15 x 25 x 35
15 x 25 x 35
15 x 35 x 25

50 x 50 x 1
75 x 75 x 1
100 x 100 x 1
150 x 150 x 1
205 x 205 x 1
280 x 280 x 1

225
1,225
2,500
5,625
10,000
22,500
42,025
78,400
reshape the image into a vector

100 x 100 x 1

10,000

reshape

10,000
what about the rest of the architecture?

how many units can we afford?

10,000 × 100 = 1 million weights
10,000 × 1,000 = 10 million weights
10,000 × 10,000 = 100 million weights
10,000 × 100,000 = 1 billion weights

how many basic patterns do we expect to find in the image?
how many patterns can the image contain?

$$100 \times 100 \times 1$$

10,000

**upper bound**

if we consider all values (1 - 256) of all 10,000 pixels, there are $256^{10,000}$ possible patterns

this is more than the number of atoms in the known, observable universe. in reality, the actual number will be much less

**lower bound**

if we want to recognize *multiple* basic, low-level patterns (e.g. edges, gradients, etc.) *anywhere in the image*, I estimate there will be a total of at least 10,000 of these.

the actual number may be far more

at least 100 million weights in the first layer alone
our approach requires a huge number of weights (parameters)

dramatically increases the amount of data/labels we need to collect

as well as the amount of computation required for training

we need to re-evaluate our approach
two sources for improving a model

- data
- priors

model

learn

know/suspect
two sources for improving a model

by observing data samples, we can learn about the data distribution

e.g. learn features common to fish and differences between them

with this knowledge, we can more easily learn mappings to/from the data to latent quantities of interest (transfer learning from unsupervised features)
two sources for improving a model

priors correspond to knowledge (or suspicions) that we already have about the task

a **data prior** is additional relevant information about a data example

label or label distribution

example/class similarity
two sources for improving a model

- **priors**
- **data priors**
- **model priors**

Priors correspond to knowledge (or suspicions) that we already have about the task.

A **model prior** is relevant information about the model/task.

- **model class/architecture**
  - depth
  - layer sizes
  - non-linearities
  - output distribution
  - activity constraints (e.g., dropout)
  - etc.

- **parameter constraints/values**
  - weight magnitude constraints (L1/L2)
  - transfer learning from a similar task
  - hand-crafted features
  - etc.
two sources for improving a model

**priors**

- **data priors**
- **model priors**

**Priors are necessary for any task**
without them, we would have no way of knowing what/how to learn

**Priors can vary in strength**
with *strong* priors, we don’t need data (we already *know* the solution)
with *weak* priors, we need a lot of data (we mostly *learn* the solution)

**Priors can be good or bad**
good/correct priors make learning easier
bad/incorrect priors make learning more difficult or impossible
two sources for improving a model

our current approach to visual object recognition relies too heavily on data

need to impose additional/stronger priors to simplify learning

we’ll impose model priors to restrict the model class for this task
properties of images

images have a notion of **locality**, which operates at **multiple scales**: neighboring *pixels* tend to be similar and vary in particular ways

![Image of pixels]

nearby *patches* tend to share characteristics and are combined in particular ways

![Image of patches]

nearby *regions* (of objects) tend to be found in particular arrangements

![Image of regions]
properties of images

what does \textit{locality} imply for our model?

more meaningful to work in image space than with reshaped vectors

units should restrict their inputs to areas of nearby units in the previous layer
properties of images

objects have a notion of **translation invariance**

Yisong’s identity is independent of his spatial location

similar statistics apply throughout the image
properties of images

what does **translation invariance** imply for our model?

the same weights should apply throughout the input

we can aggregate (**pool**) over a feature to detect whether or not it is present

can use the same weights to detect both edges

only need a single vertical edge detector to find all vertical edges

decrease the spatial size by 'summarizing' the lower-level activations

keep only a relevant summary

builds translation invariance
additional model priors - summary

work in the image-space

units only take a small window of inputs

weights will be shared across multiple units

pool each feature to create translation invariance
biological inspiration

*how do animals recognize visual stimuli?*

Hubel & Wiesel - 1950s

recorded responses of neurons in primarily visual cortex (V1) to simple visual stimuli

found neurons selective for bars at a specific orientation at specific locations

Hubel & Wiesel, 1959
biological inspiration

*how do animals recognize visual stimuli?*

**simple cells** combine lower level features (on/off ganglion responses) within a receptive field to select for more complex features.

**complex cells** combine responses from simple cells within a larger receptive field to develop translation invariance.
biological inspiration

how do animals recognize visual stimuli?

two main pathways:
recognition/what (ventral) pathway & location/where (dorsal) pathway

simple visual features are combined hierarchically to select for more complicated visual features

Kandel et al., 2012
biological inspiration

how do animals recognize visual stimuli?

areas higher in the recognition hierarchy are selective for highly specific features

these areas tend to be densely interconnected and relatively invariant to spatial location

Friewald et al., 2009, 2010
CONVOLUTIONAL NEURAL NETWORKS
(discrete) convolution

convolution is a filtering operation

convolve a filter/kernel with the input

Gaussian blur

edge detection

sharpening
(discrete) convolution

example:

$$\text{filter weights} = \begin{pmatrix} 0 & 1 & 2 \\ 2 & 2 & 0 \\ 0 & 1 & 2 \end{pmatrix}$$

take the *inner-product* of filter weights and input patches across entire input

the output is a measure of the degree of the filter feature’s presence at each location in the input, known as a **feature map**
we can **pad** the input with additional values (typically zeros) around the perimeter

this **increases** the output spatial size in comparison with non-padded convolutions

‘**same**’ padding maintains the input size

note that ‘**valid**’ padding refers to no padding

http://deeplearning.net/software/theano_versions/dev/tutorial/conv_arithmetic.html
we can apply the kernel with a *stride*, only computing the output at certain integer intervals

this decreases the output spatial size in comparison with non-strided convolutions, where the stride is one

http://deeplearning.net/software/theano_versions/dev/tutorial/conv_arithmetic.html
(discrete) convolutions

3 x 3 x 3 filter tensor

Convolutions can be applied over multiple input feature maps. In this case, instead of being a matrix, the kernel/filter is a tensor that can handle RGB images.
pooling

**example:** max pooling, 2 x 2 window, stride 2, no padding

aggregate (pool) over each feature map

pooling is performed over a window

*pad* and *stride* can also be applied

can use different operations, e.g. max, average, etc.
convolutional pop quiz

5 x 5 input feature map

3 x 3 filter

a 3 x 3 filter is convolved with a 5 x 5 input feature map

if we use unit strides and no padding (‘valid’), what is the output spatial size? 3 x 3

if we use a stride of 2 and no padding (‘valid’), what is the output spatial size? 2 x 2

if we use unit strides and half padding (‘same’), what is the output spatial size? 5 x 5

if we use a stride of 2 and half padding (‘same’), what is the output spatial size? 3 x 3
fully-connected vs. convolutional

**example calculation**

**convolutional layer** with 64 $3 \times 3$ filters operating on 3 input channels

\[ 64 \times 64 \times 3 \times 3 \times 3 + 64 \text{ biases} = 1792 \text{ parameters} \]

*note that the number of parameters is independent of the spatial size*

if the input dimension is $128 \times 128$ and the convolution is applied with unit strides and ‘same’ padding, the convolutional layer will have an output size of $64 \times 128 \times 128 = 1,048,576 \text{ units}$

**fully-connected layer** would require

\[ (128 \times 128 \times 3) \times 1,048,576 + 1,048,576 = 51,540,656,128 \text{ parameters} \]

if the assumptions we made about images (locality, translation invariance) are valid, then we have reduced the number of parameters by a factor of over *10 million*

convolutional layers allow us to trade off flexibility for an increased representation size
**image datasets**

- **MNIST**
  - 10 classes,
  - 60,000 images

- **Caltech-101**
  - 101 classes,
  - 9,146 images

- **Caltech-256**
  - 256 classes,
  - 30,607 images

- **PASCAL VOC**
  - 20 classes,
  - 9,963 images

- **CIFAR 10**
  - 10 classes,
  - 60,000 images

- **CIFAR 100**
  - 100 classes,
  - 60,000 images
image datasets

ImageNet

**Competition (ILSVRC) Dataset**
- 1,000 classes,
- 1.2 million images

**Full Dataset**
- 21,841 classes,
- 14 million images
LeNet - 1998

models
models

AlexNet - 2012
models

VGG - 2014
GoogLeNet - 2014

Inception Block:
ResNet - 2015
Inception v4 - 2016
**top-5 error on ILSVRC**

top-5 error denotes the percentage of examples for which the ground truth label is not in the model’s top 5 predictions.

1. television
2. radio
3. walkie talkie
4. pager
5. cell phone

correct top-5 prediction

Top-5 error (%)

- AlexNet: 15.3
- VGG: 7.4
- GoogLeNet: 6.7
- ResNet: 3.6
- Inception v4: 3.08
- Human: 5.1

Karpathy, 2014
number of layers

the number of layers with weights in state-of-the-art networks has grown

this is primarily due to new tricks that have been developed for training deeper networks

- AlexNet: 7
- VGG: 19
- GoogLeNet: 22
- ResNet: 34
- Inception v4: 53
number of layers

I WAS WINNING IMAGENET

UNTIL A DEEPER MODEL CAME ALONG
ensembles

as a side note:
the winning entries are typically ensembles of networks

since each network likely converges to a different local minimum,
averaging their predictions helps in generalization
data augmentation

In training these networks, people often use data augmentation to effectively boost the number of examples in the training set. We use priors on the nature of the data to create additional examples.

Example data augmentation:

- Original image
- Crops
- Rotations
- Left-right flip
what are these models learning?
filter visualization

at the first layer, we can visualize the filters as RGB images

Zeiler, 2013
for later layers, we can no longer visualize them directly in the image domain.

there are three main ways in which to visualize them:

**maximal images from dataset**
- feed in all images from the dataset and see which images make the filter activate maximally

**deconvolution**
- run the network in reverse from an intermediate layer to try to convert its activation back to the image domain

**optimized image**
- backpropagate from a filter to the image itself to find an image that would make it fire maximally
Layer 2: Top-9 Patches

- Patches from validation images that give maximal activation of a given feature map
filter visualization

Layer 2: Top-9 Patches
filter visualization

Layer 3: Top-9 Patches
filter visualization

Layer 3: Top-9 Patches
filter visualization

Layer 4: Top-9 Patches
Layer 4: Top-9 Patches
filter visualization

Layer 5. Top-9 Patches

Zeiler, 2013
filter visualization

<table>
<thead>
<tr>
<th>Layer 5: Top-9 Patches</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Layer 5: Top-9 Patches" /></td>
</tr>
</tbody>
</table>
moving beyond object recognition
object detection

R-CNN: Regions with CNN features

1. Input image
2. Extract region proposals (~2k)
3. Compute CNN features
4. Classify regions

Girshick, 2013
object detection

Faster R-CNN

Image: Diagram of Faster R-CNN architecture.
object segmentation
key point estimation
applications

handwriting recognition
- ATM
- note taking

pedestrian/object detection
for self-driving cars

face detection/recognition
- Facebook
- Microsoft
- Snapchat

search by image
- Google image search
- search by image for products on Amazon
applications

fine-grained object recognition

e.g. recognizing any bird species
applications

fine-grained object recognition

an expert birder on your phone

recognition component developed at Caltech
other cool stuff

start with an input image or random noise
randomly activate filters within the network, backpropagate to the image

depending on which filters are chosen, has different interesting effects
it's as though the network is dreaming, hence, deep dream

https://research.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html
other cool stuff

deep dream masterpieces

https://research.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html
other cool stuff

neural style transfer

start with a ‘content image’ and a ‘style image’

backpropagate the style image’s high-level statistics to the content image

Gatys et al., 2015
Johnson et al., 2016
other cool stuff

colorization

learn a mapping from black and white images to color images based on visual features
discrimination

there are patterns in images that allow us to infer latent properties

Yisong
generation
there are latent properties that result in specific patterns in images

Yisong
other cool stuff

generative modeling of images

Goodfellow et al., 2014

Goodfellow et al., 2016

Kingma et al., 2016

Radford et al., 2015

Volkhonskiy et al., 2017
conclusion

convolutions are an ideal choice when working with data with invariances
- increases parameter efficiency through model priors
- can also be applied to 1-D and 3-D data

convolutional neural networks have enabled rapid gains in nearly all areas of computer vision and other fields
- object recognition/detection/segmentation
- success depends heavily on data and hardware

many interesting new areas to explore
- work with images at multiple levels of abstraction, not just pixel level

open problems
- choosing network architecture
- limited reasoning abilities, just an input-output mapping
- learning from few examples
- understanding what/how these networks are learning, improving training
- etc.