This Lecture: Advanced Machine Learning

- **Regression Approaches**
  - Linear Regression
  - Regularization (Ridge, Lasso)
  - Kernels (Kernel Ridge Regression)
  - Gaussian Processes
- **Approximate Inference**
  - Sampling Approaches
  - MCMC
- **Deep Learning**
  - Linear Discriminants
  - Neural Networks
  - Backpropagation & Optimization
  - CNNs, RNNs, ResNets, etc.

Topics of This Lecture

- **Tricks of the Trade**
  - Recap
- **Convolutional Neural Networks**
  - Neural Networks for Computer Vision
  - Convolutional Layers
  - Pooling Layers
- **CNN Architectures**
  - LeNet
  - AlexNet
  - VGGNet
  - GoogLeNet

Recap: Choosing the Right Learning Rate

- **Convergence of Gradient Descent**
  - Simple 1D example
    \[
    W(t+1) = W(t) - \eta \frac{dE(W)}{dW}
    \]
  - What is the optimal learning rate \( \eta_{opt} \)?
    \[
    \eta_{opt} = \left( \frac{d^2E(W(t))}{dW^2} \right)^{-1}
    \]
  - Advanced optimization techniques try to approximate the Hessian by a simplified form.
  - If we exceed the optimal learning rate, bad things happen!

Recap: Advanced Optimization Techniques

- **Momentum**
  - Instead of using the gradient to change the position of the weight “particle”, use it to change the velocity.
  - Effect: dampen oscillations in directions of high curvature
    - Nesterov-Momentum: Small variation in the implementation
- **RMS-Prop**
  - Separate learning rate for each weight: Divide the gradient by a running average of its recent magnitude.
- **AdaGrad**
- **AdaDelta**
- **Adam**

Some more recent techniques, work better for some problems. Try them.

Trick: Patience

- Saddle points dominate in high-dimensional spaces!

⇒ Learning often doesn’t get stuck, you just may have to wait...
Recap: Reducing the Learning Rate

- Final improvement step after convergence is reached
  - Reduce learning rate by a factor of 10.
  - Continue training for a few epochs.
  - Do this 1-3 times, then stop training.

- Effect
  - Turning down the learning rate will reduce the random fluctuations in the error due to different gradients on different minibatches.

- Be careful: Do not turn down the learning rate too soon!
  - Further progress will be much slower after that.

Topics of This Lecture

- Tricks of the Trade
  - Recap
  - Convolutional Neural Networks
    - Neural Networks for Computer Vision
    - Convolutional Layers
    - Pooling Layers
  - CNN Architectures
    - LeNet
    - AlexNet
    - VGGNet
    - GoogLeNet

Neural Networks for Computer Vision

- How should we approach vision problems?

- Architectural considerations
  - Input is 2D: 2D layers of units
  - No pre-segmentation: Need robustness to misalignments
  - Vision is hierarchical: Hierarchical multi-layered structure
  - Vision is difficult: Network should be deep

Why Hierarchical Multi-Layered Models?

- Motivation 1: Visual scenes are hierarchically organized

- Motivation 2: Biological vision is hierarchical, too

**Inspiration: Neuron Cells**

Object → Face Y/N?

Object → Object parts → Primitive features → Input image

Face → Eyes, nose, ... → Oriented edges → Face image

Interotemporal cortex → V4: different textures → V1: simple and complex cells → Photoreceptors, retina

Axon from another cell → Axonal arborization → Synapse → Dendrite → Axon → Cell body or Soma → Synapses

Slide adapted from Geoff Hinton

Slide adapted from Geoff Hinton

B. Leibe

Slide adapted from Richard Turner

B. Leibe

Slide adapted from Richard Turner

B. Leibe

Slide adapted from Richard Turner

B. Leibe
Hubel/Wiesel Architecture

  - Visual cortex consists of a hierarchy of simple, complex, and hyper-complex cells

Why Hierarchical Multi-Layered Models?

- Motivation 3: Shallow architectures are inefficient at representing complex functions

What’s Wrong With Standard Neural Networks?

- Complexity analysis
  - How many parameters does this network have?
    \[ |θ| = 3D^2 + D \]
  - For a small 32x32 image
    \[ |θ| = 3 \times 32^4 + 32^2 \approx 3 \times 10^9 \]
- Consequences
  - Hard to train
  - Need to initialize carefully
- **Convolutional nets reduce the number of parameters!**

Convolutional Networks (CNN, ConvNet)

- Neural network with specialized connectivity structure
  - Stack multiple stages of feature extractors
  - Higher stages compute more global, more invariant features
  - Classification layer at the end

Convolutional Networks: Intuition

- Fully connected network
  - E.g. 1000x1000 image
  - 1M hidden units
  - \( 1T \) parameters!

- Locally connected net
  - E.g. 1000x1000 image
  - 1M hidden units
  - 10x10 receptive fields
  - \( 100M \) parameters!
Convolutional Networks: Intuition

- Convolutional net
  - Share the same parameters across different locations
  - Convolutions with learned kernels

Learn multiple filters
- E.g. 1000 x 1000 image
  - 100 filters
  - 10 x 10 filter size
  ⇒ 10k parameters

Result: Response map
- size: 1000 x 1000 x 100
  - Only memory, not params!

- Learn multiple filters
  - E.g. 1000 x 1000 image
  - 100 filters
  - 10 x 10 filter size
  ⇒ 10k parameters

Result: Response map
- size: 1000 x 1000 x 100
  - Only memory, not params!

Important Conceptual Shift

- Before
- Now:

Convolution Layers

- Note: Connectivity is
  - Local in space (5 x 5 inside 32 x 32)
  - But full in depth (all 3 depth channels)

Naming convention:
Convolution Layers

Example: 7x7 input
assume 3x3 connectivity
stride 1

• Replicate this column of hidden neurons across space,
  with some stride.

Slide credit: FeiFei Li, Andrej Karpathy
B. Leibe

Example: 7x7 input
assume 3x3 connectivity
stride 1

• Replicate this column of hidden neurons across space,
  with some stride.

Example: 7x7 input
assume 3x3 connectivity
stride 1

⇒ 5x5 output

• Replicate this column of hidden neurons across space,
  with some stride.

What about stride 2?
Convolution Layers

- Replicate this column of hidden neurons across space, with some stride.
- In practice, common to zero-pad the border.
  - Preserves the size of the input spatially.

Example:
7 x 7 input
assume 3 x 3 connectivity
stride 1
⇒ 5 x 5 output

What about stride 2?

Activation Maps of Convolutional Filters

- Each activation map is a depth slice through the output volume.

Effect of Multiple Convolution Layers

Convolutional Networks: Intuition

- Let’s assume the filter is an eye detector
  - How can we make the detection robust to the exact location of the eye?

Image source: Yann LeCun
Convolutional Networks: Intuition

- Let’s assume the filter is an eye detector
  - How can we make the detection robust to the exact location of the eye?

- Solution:
  - By pooling (e.g., max or avg) filter responses at different spatial locations, we gain robustness to the exact spatial location of features.

Max Pooling

- Effect:
  - Make the representation smaller without losing too much information
  - Achieve robustness to translations

Max Pooling

- Note
  - Pooling happens independently across each slice, preserving the number of slices.

CNNs: Implication for Back-Propagation

- Convolutional layers
  - Filter weights are shared between locations
  - Gradients are added for each filter location.

Topics of This Lecture

- Tricks of the Trade
  - Recap

- Convolutional Neural Networks
  - Neural Networks for Computer Vision
  - Convolutional Layers
  - Pooling Layers

- CNN Architectures
  - LeNet
  - AlexNet
  - VGGNet
  - GoogleNet


- Early convolutional architecture
  - 2 Convolutional layers, 2 pooling layers
  - Fully-connected NN layers for classification
  - Successfully used for handwritten digit recognition (MNIST)
ImageNet Challenge 2012

- ImageNet
  - ~14M labeled internet images
  - 20k classes
  - Human labels via Amazon Mechanical Turk

- Challenge (ILSVRC)
  - 1.2 million training images
  - 1000 classes
  - Goal: Predict ground-truth class within top-5 responses
  - Currently one of the top benchmarks in Computer Vision

CNN Architectures: AlexNet (2012)

- Similar framework as LeNet, but
  - Bigger model (7 hidden layers, 650k units, 60M parameters)
  - More data ($10^6$ images instead of $10^3$)
  - GPU implementation
  - Better regularization and up-to-date tricks for training (Dropout)

ImageNet Classification with Deep Convolutional Neural Networks, NIPS 2012.

ILSVRC 2012 Results

- AlexNet almost halved the error rate
  - 16.4% error (top-5) vs. 26.2% for the next best approach
  - A revolution in Computer Vision
  - Acquired by Google in Jan ‘13, deployed in Google+ in May ‘13

AlexNet Results

- CNN Architectures: VGGNet (2014/15)

AlexNet Results

- Test image
- Retrieved images

CNN Architectures: VGGNet (2014/15)

- **Main ideas**
  - Deeper network
  - Stacked convolutional layers with smaller filters (+ nonlinearity)
  - Detailed evaluation of all components

- **Results**
  - Improved ILSVRC top-5 error rate to 6.7%.

<table>
<thead>
<tr>
<th>Layer</th>
<th>VGG-16</th>
<th>VGG-19</th>
<th>VGG-19-A</th>
<th>VGG-19-B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conv2D</td>
<td>11 weight layers</td>
<td>19 weight layers</td>
<td>25 weight layers</td>
<td>39 weight layers</td>
</tr>
<tr>
<td>Conv2D</td>
<td>11 weight layers</td>
<td>19 weight layers</td>
<td>25 weight layers</td>
<td>39 weight layers</td>
</tr>
<tr>
<td>Conv2D</td>
<td>11 weight layers</td>
<td>19 weight layers</td>
<td>25 weight layers</td>
<td>39 weight layers</td>
</tr>
<tr>
<td>Conv2D</td>
<td>11 weight layers</td>
<td>19 weight layers</td>
<td>25 weight layers</td>
<td>39 weight layers</td>
</tr>
<tr>
<td>Output</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

GoogLeNet Visualization

- Inception module + copies
- Auxiliary classification outputs for training the lower layers (deprecated)

CNN Architectures: GoogLeNet (2014)

- **Main ideas**
  - "Inception" module as modular component
  - Learns filters at several scales within each module


Results on ILSVRC

<table>
<thead>
<tr>
<th>Method</th>
<th>top-1 val. error (%)</th>
<th>top-5 val. error (%)</th>
<th>top-5 test error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>VGG (2 sets, multi-crop &amp; data evol.)</td>
<td>25.7</td>
<td>7.1</td>
<td>7.0</td>
</tr>
<tr>
<td>VGG (2 sets, single-crop &amp; data evol.)</td>
<td>25.7</td>
<td>7.1</td>
<td>7.0</td>
</tr>
<tr>
<td>GoogLeNet (Szegedy et al., 2014/15)</td>
<td>6.9</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Mina (Ioffe et al., 2014/11/15)</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ResNet (He et al., 2014/11/15)</td>
<td>6.9</td>
<td>6.9</td>
<td>6.9</td>
</tr>
<tr>
<td>ResNet (He et al., 2015/11/15)</td>
<td>3.9</td>
<td>3.9</td>
<td>3.9</td>
</tr>
<tr>
<td>Inception (Szegedy et al., 2015/11/15)</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Inception (Szegedy et al., 2016/11/15)</td>
<td>3.9</td>
<td>3.9</td>
<td>3.9</td>
</tr>
<tr>
<td>Inception (Szegedy et al., 2017/11/15)</td>
<td>3.9</td>
<td>3.9</td>
<td>3.9</td>
</tr>
</tbody>
</table>

References and Further Reading

- **LeNet**

- **AlexNet**

- **VGGNet**
  - K. Simonyan, A. Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition, ICLR 2015

- **GoogLeNet**

References

- **ReLU**

- **Batch Normalization**