Fundamentals of Deep Learning

Raymond Ptucha,
Rochester Institute of Technology
NVIDIA, Deep Learning Institute

Electronic Imaging 2019: SC20
January 15, 2019, 8:30am-12:45pm

Fair Use Agreement

This agreement covers the use of all slides in this document, please read carefully.

- You may freely use these slides, if:
  - You send me an email telling me the conference/venue/company name in advance, and which slides you wish to use.
  - You receive a positive confirmation email back from me.
  - My name (R. Ptucha) appears on each slide you use.

(c) Raymond Ptucha, rwpecc@rit.edu
Agenda

• Part I- Intuition and Theory  
  – 8:35-9:15pm: Introduction  
  – 9:15-10:00pm: Convolutional Neural Networks  
  – 10:00-10:40pm: Recurrent Neural Networks  

• 10:40-11:00pm: Break  

• Part II- Hands on  
  – 11:00am-12:45pm: Hands-on exercises

Machine Learning

• Machine learning is giving computers the ability to analyze, generalize, think/reason/behave like humans.  

• Machine learning is transforming medical research, financial markets, international security, and generally making humans more efficient and improving quality of life.  

• Inspired by the mammalian brain, deep learning is machine learning on steroids- bigger, faster, better!
The point of Singularity

• The point of singularity is when computers become smarter than humans.

Unleashing of Intelligence

• Machines will slowly match, then quickly surpass human capabilities.
• Today it is exciting/scary/fun to drive next to an autonomous car.
• Tomorrow it may be considered irresponsible for a human to relinquish control from a car that has faster reaction times, doesn’t drink/text/get distracted/tired, and is communicating with surrounding vehicles and objects.
Why is AI (Deep Learning) Just Now Becoming Practical in Many Day-to-Day Situations?

- Availability of data;
- Sustained advances in hardware capabilities (including GPUs running machine learning workloads);
- Omnipresent connectivity;
- Lower cost and power consumption;
- Sustained advances in algorithmic learning techniques.

2017: The Year of AI:
The Wall Street Journal, Forbes, and Fortune

NEC Face Recognition

SONY Playstation Virtual Reality

Evolutionary Reinforcement Learning
2017: The Year of AI:
The Wall Street Journal, Forbes, and Fortune

DeepBach

NVIDIA Autonomous Car Detection & Segmentation

YOLO v2 Object Detection

Awesome 2018 Technology

Awesome 2018 Technology

<table>
<thead>
<tr>
<th>Faceshift GDC</th>
<th>Apple iPhone X, Animoji Yourself</th>
</tr>
</thead>
</table>

R. Ptucha ‘19

Awesome 2018 Technology

NVIDIA DRIVE
Autonomous Vehicle Platform
October 16, 2017

NVIDIA Drive

R. Ptucha ‘19
Awesome 2018 Technology

Email smart compose sentence completion


ML Trends

• If 2013/2014 were the year of CNNs, 2015/2016 were LSTMs, 2018 was GANs.
• In 2017 AI fear mongering peaked, in 2018 press has come to terms with AI limitations.
• In 2018 focus started to turn to fairness, bias, interpretability.
• Deep learning continues to infiltrate more applications/markets.
• While current deep learning is not capable of achieving general AI, it may form building blocks.
The Human Brain

- We’ve learned more about the brain in the last 5 years than we have learned in the last 5000 years!
- It controls every aspect of our lives, but we still don’t understand exactly how it works.

The Brain on Pattern Recognition

- Airplane, Cat, Car, Dog

STL-10 dataset

http://thebraingeek.blogspot.com/2012/08/blindsight.html
The Brain on Pattern Recognition

Despite Changes in Deformation:

The Brain on Pattern Recognition

Despite Changes in Occlusion:
Despite Changes in Size, Pose, Angle:

Tardar Sauce "Grumpy Cat"

Despite Changes in Background Clutter:

R. Ptucha '19
The Brain on Pattern Recognition

Despite Changes in Class Variation…

Teaching Computers to See

- It took evolution 540M years to develop the marvel of the eye-brain.
- Let’s say a child collects a new image every 200msec.
- By age 3, this child has processed over 100M images.

\[(5 \text{ images/sec})(60 \text{ sec/min})(60 \text{ min/hr})(24 \text{ hr/day})(365 \text{ days/yr})(3 \text{ yrs}) = 236M\]

- Today’s computers can do this in a few days…
Neural Nets on Pattern Recognition

- Instead of trying to code simple intuitions/rules on what makes an airplane, car, cat, and dog…
- We feed neural networks a large number of training samples, and it will automatically learn the rules!
- We will learn the magic behind this today!

Artificial Neuron

Note, $x_0$ is the bias unit, $x_0=1$

$$h_\theta(x) = g(x_0\theta_0 + x_1\theta_1 + ... + x_n\theta_n) = g\left(\sum_{i=0}^{n} x_i\theta_i\right)$$

$$h_\theta(x) = g(\theta^T x)$$
Artificial Neural Networks

- **Artificial Neural Network (ANN)** – A network of interconnected nodes that “mimic” the properties of a biological network of neurons.

![Diagram of an Artificial Neural Network](image)

4-Layer ANN Fully Connected Topology

- **Input Layer**
- **Hidden Layer 1**
- **Hidden Layer 2**
- **Output Layer**

A 20x20 image would have 400 input nodes, **a** nodes in Hidden Layer 1, **b** nodes in Hidden Layer 2, and **C** nodes in the Output Layer, where **C** is the number of classes.

Backpropagation (~1985) uses \( \frac{\partial \Delta}{\partial w} \) for learning. Learning happens in the weights - each line is a weight.

R. Ptucha ’19
Basic operation of an artificial neuron involves summing its weighted input signal and applying a threshold, called an activation function.

- If the sum is greater than threshold, fire, otherwise don’t.
- A linear activation function is unbounded and limits the nonlinear properties of our net.
- Non-linear hyperbolic tangent shown.

**Sigmoid:**
\[ h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}} \]

**Hyperbolic tangent:**
\[ h_\theta(x) = \frac{e^{\theta^T x} - e^{-\theta^T x}}{e^{\theta^T x} + e^{-\theta^T x}} \]
Activation Function Comparison

- **Tanh**
- **Sigmoid**
  - Gradient of both saturates at zero. Sigmoid also non-zero centered, so in practice tanh performs better.

- **Rectified Linear Units (ReLU)**
  - Better for high dynamic range
  - Faster learning
  - Overall better result
  - Neurons can “die” if allowed to grow unconstrained

Tanh vs. ReLU on CIFAR-10 dataset [Krizhevsky’12]

ReLU reaches 25% error 6× faster!

Note: Learning rates optimized for each, no regularization, four layer CNN.
### Where Do Weights Come From?

- The weights in a neural network need to be learned such that the errors are minimized.
- Just like logistic regression, we can write a cost function.
- Similar to gradient descent, we can write an iterative procedure to update weights, with each iteration decreasing our cost.
- These iterative methods may be less efficient than a direct analytical solution, but are easier to generalize.

---

### Backpropagation

- We need to solve weights of a network so that the error is minimized.
- Weights can be refined by changing each weight by an amount proportional to the partial derivative of the error with respect to each weight.
- Partial derivatives can be calculated by iteratively changing each weight and measuring the corresponding change in error.
- Hard to do with millions of weights!
Hyperparameters vs. parameters

- Hyperparameters are the tuning parameters for a nnet—say number of layers, nodes per layer, learning rate, momentum, regularization, etc.
- Parameters are the weights that are being learned. Ignoring bias terms, the below network has 3x5+5x5+5x4 = 60 parameters.
  - If we include bias terms, we have 4x5+6x5+6x4=74 learnable parameters.

Note: deep nets may contain 100M parameters with 20 layers!

Cost Function

Logistic Regression Cost Function:

\[ J(\theta) = -\frac{1}{n} \sum_{i=1}^{n} y^{(i)} \log \left( h_{\theta}(x^{(i)}) \right) + (1 - y^{(i)}) \log \left( 1 - h_{\theta}(x^{(i)}) \right) \]

\[ n \] is the number of training samples, \( D \) is the dimension of each sample

Neural Network Cost Function:

\[ J(\theta) = -\frac{1}{n} \sum_{i=1}^{n} \sum_{c=1}^{C} y^{(i,c)} \log \left( h_{\theta}(x^{(i)}) \right) + (1 - y^{(i,c)}) \log \left( 1 - h_{\theta}(x^{(i)}) \right) \]

\[ \frac{\lambda}{2n} \sum_{j=1}^{D} \theta_j^2 \]

\( C \) is the number of classes
\( L \) is the number of layers
\( s_l \) is the number of nodes in layer \( l \)

Add regularization to prevent overfitting to the training set.
Initialization of Weights

- We initialize weights to small ± values centered on 0.
- If they were all initialized to 0, the network wouldn’t learn anything.
  – i.e: the output of each node would be equal, therefore the update would update each term identically.
- If they were all initialized to large values (values > +1 or < -1), the inputs to each node would be saturated.
- If they were all initialized to small values around 0, we would be in the linear portion of the activation function.
  – \( W_1 = 0.001 \times \text{rand} (\text{length}(h_1), \text{length}(h_2)) \)

Although the uniform distribution is good, the more inputs to a node, the greater its variance.

To set output distribution of all nodes equal (this empirically improves convergence), use

:\[ \epsilon_{\text{init}} = \frac{\sqrt{6}}{\sqrt{s_l + s_{l+1}}} \]

\( s_l, s_{l+1} \) are the No. nodes in layers around \( \theta^0 \)

- Other solutions restrict weights: \(-\epsilon_{\text{init}} < b^*_i < \epsilon_{\text{init}}\)

For deep ReLU networks, He2015 showed:

- \( W_1 = 0.001 \times \text{rand}(\text{length}(h_1), \text{length}(h_2)) / \sqrt{2 \times \text{length}(h_1)} \)
- Works best and is recommended for them
Multiclass Loss Functions

- The input image scores highest against cat, but is also somewhat similar to dog.
- How do we assign a loss function?

Activation Function of Output Layer

- Sigmoid returns 0 or 1 for each output node.
- What if you wanted a confidence interval?
- Use a linear activation function for regression: $a_l^{(l)} = z_l^{(l)}$
- Softmax often used for classification:

$$a^{(l)}_c = h_\theta(x)_c = g\left(z^{(l)}_c\right) = \frac{\exp\left(z^{(l)}_c\right)}{\sum_{c'=1,c}^C \exp\left(z^{(l)}_{c'}\right)}$$

- Note: Only the output layer activation function changes- all hidden layer nodes activation functions would be the sigmoid/tanh/ReLU function.
20

R. Ptucha '19

50

exp() take in values +/- values and returns all positive values.

Most Common Loss Functions

• The cost function we previously used was a direct copy from logistic regression and works great for binary classification.
• For multi-class classification, two popular loss methods:
  1. Cross-entropy loss, which uses softmax
     \[ \text{Loss}(i) = -\sum_{c=1}^{C} \text{exp}(z_c^{(i)}) \times \log\left(\frac{\text{exp}(z_c^{(i)})}{\sum_{c=1}^{C} \text{exp}(z_c^{(i)})}\right) \times 100 \]
  where \( z_c^{(i)} = \log\left(\frac{\text{exp}(z_c^{(i)})}{\sum_{c=1}^{C} \text{exp}(z_c^{(i)})}\right) \times 100 \)

• Multiclass SVM Loss (Weston Watkins formulation):

R. Ptucha '19

51
Most Common Loss Functions

- The cost function we previously used was a direct copy from logistic regression and works great for binary classification.
- For multi-class, there are two popular data loss methods:
  1. Cross-entropy loss, which uses softmax:

\[
Loss^{(i)} = -\log \left( \frac{\exp\left(\text{out}^{(i)}_{y_i}\right)}{\sum_{c=1}^{C} \exp\left(\text{out}^{(i)}_{c}\right)} \right)
\]

   Loss for sample \( i \) = \( \exp(\text{output of GT node}) \) / \( \sum \exp(\text{output of all nodes}) \)

  2. Multiclass SVM Loss (Weston Watkins formulation):

\[
Loss^{(i)} = \sum_{j \neq y_i} \max\left(0, \text{out}_j - \text{out}_{y_i} + \Delta\right)
\]

   Sum of incorrect – correct classes

Loss Example
(based on cs231n, Li/Karpathy 2016)
Local Minima

- Back propagation modifies the weights to minimize the cost function.
- It does this by approximating the gradient of the error and following it downhill.
- We could of course get stuck at a local minima.
- If we start near the global minimum, we often end up there.
- If we start at a local minimum, we often end up there!

Learning Rates

- Like gradient descent, we incorporate a learning rate so we don’t take full steps in any one direction.
- In its simplest form, we have $w_{ij} = w_{ij} - \eta \, dw_{ij}$
- As training progresses, we generally anneal the learning rate over time, going from large to small steps. Three common approaches:
  1. Step decay: Either watch validation error and reduce learning rate whenever validation error stops improving, or automatically reduce by $\sim 0.1$ every $\sim 20$ epochs
  2. Exponential decay: $\alpha = \alpha_0 \, e^{-kt}$ ($\alpha_0$ and $k$ are hyperparameters, $t$ is epoch).
  3. $1/t$ decay: $\alpha = \alpha_0 / (1 + kt)$ ($\alpha_0$ and $k$ are hyperparameters, $t$ is epoch).
Momentum

- One solution is to train several networks, each at a different initialization.
- Another is to add a momentum term.
- Imagine a ball rolling down a hill- if it had little energy, it may get stuck in little dips.
- If it had a lot of energy, it would be more likely to skip over little dips, and keep rolling until it hopefully hits a global minimum.

Adding a momentum term can sometimes avoid local minima and make training faster.

Momentum

- Momentum adds a contribution of our previous weight change to our current weight change.

\[ \Delta w_j^{(l)} = \Delta w_j^{(l)} + \alpha \Delta w_j^{(l)} \delta_j^{(l+1)} \]

- To:

\[ \Delta w_j^{(l)}_t = \Delta w_j^{(l)}_{t-1} + \alpha \Delta w_j^{(l)}_{t-1} + \alpha \Delta w_j^{(l)}_{t-1} , 0 \leq \alpha \leq 1 \]

Caution- these methods are quite effective, but more of an art than science!

The use of \( t \) and \( t-1 \) emphasize the previous and current derivatives.
A typical value of \( \alpha \) is 0.9.
Sequential vs. Batch Training

- Back propagation can be done:
  - Sequential: one sample at a time,
    - Weights are shifted back and forth quite a bit
  - Group (minibatch): a group of samples at a time, or
  - Batch: all training samples at once
    - Weights are shifted in direction that makes most input samples better
    - Generally converges the fastest
- Recommended to use largest minibatch possible and stay within memory constraints of hardware.

Examples of Learning Rate and Batch Size

This batch size could be made a little larger to shrink the variance
Tuning Parameters

- Curve too linear—increase learning rate.
- Train and Test sets too similar—increase model complexity.
- Be careful, larger models susceptible to overfitting.
  Perhaps increase regularization.

Bias (underfit) vs. Variance (overfit) errors

- Model too simple: Too high error on train and test
- Model too complex: Overfitting to training set

Adopted from: Andrew Ng, ML class
Regularization Tuning

\[ J_{\text{cv}}(\theta) \]
\[ J_{\text{train}}(\theta) \]

Sweet spot!

More regularization → error

Adopted from:
Andrew Ng, ML class

R. Ptucha '19

For More Information: [http://www.rit.edu/mil](http://www.rit.edu/mil)

Raymond W. Ptucha
Assistant Professor, Computer Engineering
Director, Machine Intelligence Laboratory
Rochester Institute of Technology

Email: rwpeece@rit.edu
Phone: +1 (585) 797-5561
Office: GLE (09) 3441

R. Ptucha '19