Lecture 2:
Deep Learning Basics

Instructor: Hao Su
Jan 11, 2018
Agenda

• Motivation of Building Deeper Networks

• Ideas in Deep Net Architectures

• Deep Learning Practice (Vignesh Gokul)
Neural Network: A Compositional Function

**Model:** Multi-Layer Perceptron (MLP)

**Loss function:** L2 loss

**Optimization:** Gradient descent

\[ y' = W_3f(W_2f(W_1x + b_1) + b_2) + b_3 \]

\[ l(y, y') = (y - y')^2 \]

\[ W = W - \eta \frac{\partial L}{\partial W} \]
Universal Approximation Theorem

A three-layer network approximates any continuous function

Let \( \varphi(\cdot) \) be a nonconstant, bounded, and monotonically-increasing continuous function. Let \( I_m \) denote the \( m \)-dimensional unit hypercube \([0, 1]^m\). The space of continuous functions on \( I_m \) is denoted by \( C(I_m) \). Then, given any function \( f \in C(I_m) \) and \( \varepsilon > 0 \), there exists an integer \( N \), real constants \( v_i, b_i \in \mathbb{R} \) and real vectors \( w_i \in \mathbb{R}^m \), where \( i = 1, \cdots, N \), such that we may define:

\[
F(x) = \sum_{i=1}^{N} v_i \varphi\left(w_i^T x + b_i\right)
\]

as an approximate realization of the function \( f \) where \( f \) is independent of \( \varphi \); that is,

\[
|F(x) - f(x)| < \varepsilon
\]

for all \( x \in I_m \). In other words, functions of the form \( F(x) \) are dense in \( C(I_m) \).
Universal Approximation Theorem

A three-layer network approximates any continuous function

Let \( \varphi(\cdot) \) be a nonconstant, bounded, and monotonically-increasing continuous function. Let \( I_m \) denote the \( m \)-dimensional unit hypercube, \([0, 1]^m\). The space of functions of all \( I_m \) is denoted by \( C(I_m) \). Then, given any function \( f \in C(I_m) \), where \( i = 1, \ldots, m \),

\[
F(x) = \sum_{i=1}^{N} w_i \varphi(x - \theta_i)
\]

as an approximate, where

\[
|F(x) - f(x)| < \epsilon
\]

for all \( x \in I_m \). In other words, functions of the form \( F(x) \) are dense in \( C(I_m) \).
Overfitting is correlated with the complexity of learning model

In exponential family, Bayesian Information Criterion (BIC) for Model Selection

$$-2 \cdot \ln p(x \mid M) \approx \text{BIC} = -2 \cdot \ln \hat{L} + k \cdot \ln(n) + O(1)$$

- $\hat{L}$ = the maximized value of the likelihood function of the model, i.e., where $\hat{\theta}$ are the parameter values that maximize the likelihood function;
- $x$ = the observed data;
- $n$ = the number of data points in $x$, the number of observations, or equivalently, the sample size;
- $k$ = the number of parameters estimated by the model.
A Principle in Learning Algorithm Design

- Overfitting is correlated with the complexity of learning model

In exponential family, Bayesian Information Criterion (BIC) for Model Selection

\[ -2 \cdot \ln p(x \mid M) \approx \text{BIC} = -2 \ln \hat{L} + k \ln(n) + O(1) \]

Smaller is better

- \( \hat{L} \) = the maximized value of the likelihood function of the model, i.e., where \( \hat{\theta} \) are the parameter values that maximize the likelihood function;
- \( x \) = the observed data;
- \( n \) = the number of data points in \( x \), the number of observations, or equivalently, the sample size;
- \( k \) = the number of parameters estimated by the model.
A Principle in Learning Algorithm Design

• Overfitting is correlated with the complexity of learning model

In exponential family, Bayesian Information Criterion (BIC) for Model Selection

\[-2 \cdot \ln p(x \mid M) \approx \text{BIC} = -2 \cdot \ln \hat{\mathcal{L}} + k \cdot \ln(n) + O(1)\]

• \(\hat{\mathcal{L}}\) = the maximized value of the likelihood function of the model, i.e., where \(\hat{\theta}\) are the parameter values that maximize the likelihood function;
• \(x\) = the observed data;
• \(n\) = the number of data points in \(x\), the number of observations, or equivalently, the sample size;
• \(k\) = the number of parameters estimated by the model.
A Principle in Learning Algorithm Design

- Overfitting is correlated with the complexity of learning model

Probably Approximately Correct (PAC) theory

\[ \Pr \left( \text{test error} \leq \text{training error} + \sqrt{\frac{1}{N} \left[ D \left( \log \left( \frac{2N}{D} \right) + 1 \right) - \log \left( \frac{\eta}{4} \right) \right]} \right) = 1 - \eta, \]

where \( D \) is the VC dimension of the classification model, \( 0 \leq \eta \leq 1 \), and \( N \) is the size of the training set (restriction: this formula is valid when \( D \ll N \). When \( D \) is larger, the test-error may be much higher than the training-error. This is due to overfitting).

[from Wikipedia]
Occam’s Razor Principle

Entia non sunt multiplicanda praeter necessitatem.

William of Ockham, 14th century

Suppose there exist two explanations for an occurrence. In this case the simpler one is usually better.
The Intuition behind Pushing Deep

![Diagram of a neural network](image)

\[ \sum a_i = \text{ReLU}(x - \theta_i) \]

\[ \text{ReLU}(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases} \]
The Intuition behind Pushing Deep

\[ a_i = \text{ReLU}(x - \theta_i) \]

\[ y = \sum_i \text{ReLU}(x - \theta_i) \]

ReLU(x) = \begin{cases} 
0 & \text{if } x < 0 \\
\text{x} & \text{if } x \geq 0 
\end{cases}

piece-wise linear, 6 knots
The Intuition behind Pushing Deep

\[ a_{1,i} = \text{ReLU}(x - \theta_i) \quad a_{2,i} = \text{ReLU}(x - \phi_i) \quad y \]
The Intuition behind Pushing Deep

\[
a_{1,i} = \text{ReLU}(x - \theta_i) \quad a_{2,i} = \text{ReLU}(x - \phi_i)
\]

\[
y = \sum_{1 \leq i \leq 3} \text{ReLU}([\sum_{1 \leq j \leq 3} \text{ReLU}(x - \theta_j)] - \phi_i)
\]

piece-wise linear, can have 9 knots!
The Intuition behind Pushing Deep

Interpretation I: With the same number of parameters, create combinatorial data flow
The Intuition behind Pushing Deep

Interpretation I: With the same number of parameters, create combinatorial data flow
Interpretation II: Abstract data progressively
Alexnet
'Machine Learning is the new electricity.'
- Andrew Ng

“Machine Learning has become alchemy.”
- Ali Rahimi (at NIPS 2017)
IDEAS IN DEEP NET ARCHITECTURES
What people think I am doing when I “build a deep learning model”

What I actually do...
Contents

• **Building blocks:** fully connected, ReLU, conv, pooling,

• **Classic architectures:** MLP, LeNet, AlexNet, VGG, ResNet
Multi-Layer Perceptron

Fully Connected

http://playground.tensorflow.org/
- The first learning machine: the **Perceptron** Built at Cornell in 1960

- The Perceptron was a (binary) linear classifier on top of a simple feature extractor

\[ y = \text{sign} \left( \sum_{i=1}^{N} W_i F_i(X) + b \right) \]
Non-linear Op

Sigmoid
\[ \sigma(x) = \frac{1}{1 + e^{-x}} \]

Tanh
\[ \tanh(x) = 2\sigma(2x) - 1 \]

Major drawbacks: Sigmoids saturate and kill gradients

From CS231N
ReLU (Rectified Linear Unit)

\[ f(x) = \max(0, x) \]

- Cheaper (linear) compared with Sigmoid (exp)
- No gradient saturation, faster in convergence
- “Dead” neurons if learning rate set too high

Other Non-linear Op:
Leaky ReLU, \( f(x) = 1(x < 0)(\alpha x) + 1(x \geq 0)(x) \)
MaxOut \( \max(w_1^T x + b_1, w_2^T x + b_2) \)

A plot from Krizhevsky et al. paper indicating the 6x improvement in convergence with the ReLU unit compared to the tanh unit.

From CS231N
Convolutional Neural Network: LeNet (1998 by LeCun et al.)

One of the first successful applications of CNN.

(pooling)
Fully Connected NN in high dimension

Example: 200x200 image
- Fully-connected, 400,000 hidden units = 16 billion parameters
- Locally-connected, 400,000 hidden units 10x10 fields = 40 million params
- Local connections capture local dependencies

Shared Weights & Convolutions: Exploiting Stationarity

Example: 200x200 image
- 400,000 hidden units with 10x10 fields = 1000 params
- 10 feature maps of size 200x200, 10 filters of size 10x10

Slide from LeCun
Convolution

Stride 1

-2  2  1  2  1
0  1  2  -1  1  -3  0

Stride 2

-2  1  1
0  1  2  -1  1  -3  0

Pad 1

Stride 2

From CS231N

Pad 1

Stride 1

From vdumoulin/conv_arithmetic
Discarding pooling layers has been found to be important in training good generative models, such as variational autoencoders (VAEs) or generative adversarial networks (GANs).

It seems likely that future architectures will feature very few to no pooling layers.

*From CS231N*
LeNet (1998 by LeCun et al.)

- Fully Connected
- Convolution
- Non-linear Op
- Pooling
AlexNet (2012 by Krizhevsky et al.)

- 8 layers: first 5 convolutional, rest fully connected
- ReLU nonlinearity
- Local response normalization
- Max-pooling
- Dropout

Source: [Krizhevsky et al., 2012]
AlexNet (2012 by Krizhevsky et al.)

- Zero every neuron with probability $1 - p$
- At test time, multiply every neuron by $p$

Source: [Srivastava et al., 2014]
AlexNet (2012 by Krizhevsky et al.)

- Stochastic gradient descent
- Mini-batches
- Momentum
- Weight decay ($\ell_2$ prior on the weights)

Filters trained in the first layer

Source: [Krizhevsky et al., 2012]
- The number of training examples is 1.2 million
- The number of parameters is 5-155 million
- How does the network manage to generalize?
• Deeper than AlexNet: 11-19 layers versus 8
• No local response normalization
• Number of filters multiplied by two every few layers
• Spatial extent of filters $3 \times 3$ in all layers
• Instead of $7 \times 7$ filters, use three layers of $3 \times 3$ filters
  • Gain intermediate nonlinearity
  • Impose a regularization on the $7 \times 7$ filters
Formally, deeper networks contain shallower ones (i.e. consider no-op layers)

**Observation:** Deeper networks not always lower training error

**Conclusion:** Optimization process can’t successfully infer no-op
• Solves problem by adding skip connections
• Very deep: 152 layers
• No dropout

Batch normalization

Source: Deep Residual Learning for Image Recognition
Algorithm 2 Batch normalization [Ioffe and Szegedy, 2015]

**Input:** Values of $x$ over minibatch $x_1 \ldots x_B$, where $x$ is a certain channel in a certain feature vector

**Output:** Normalized, scaled and shifted values $y_1 \ldots y_B$

1: $\mu = \frac{1}{B} \sum_{b=1}^{B} x_b$

2: $\sigma^2 = \frac{1}{B} \sum_{b=1}^{B} (x_b - \mu)^2$

3: $\hat{x}_b = \frac{x_b - \mu}{\sqrt{\sigma^2 + \epsilon}}$

4: $y_b = \gamma \hat{x}_b + \beta$

- Accelerates training and makes initialization less sensitive
- Zero mean and unit variance feature vectors
ResNet (2016 by Kaiming He et al.)

152 layers

ILSVRC'15
ResNet

ILSVRC'14
GoogleNet

ILSVRC'14
VGG

ILSVRC'13
8 layers

ILSVRC'12
AlexNet

ILSVRC'11
shallow

ILSVRC'10

[He et al., 2016]

[Donoho et al, STATS385]