Big Data Mining

Convolutional Neural Networks (CNN)

Min-Yuh Day, Ph.D.
Assistant Professor
Department of Information Management
Tamkang University

1071BDM11
TLVXM1A (M2244) (8619) (Fall 2018)
(MBA, DBETKU) (3 Credits, Required) [Full English Course]
(Master’s Program in Digital Business and Economics)
Mon, 9, 10, 11, (16:10-19:00) (B206)

http://mail.tku.edu.tw/myday
2018-12-03
<table>
<thead>
<tr>
<th>Week</th>
<th>Date</th>
<th>Subject/Topics</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2018/09/10</td>
<td>Course Orientation for Big Data Mining</td>
</tr>
<tr>
<td>2</td>
<td>2018/09/17</td>
<td>ABC: AI, Big Data, Cloud Computing</td>
</tr>
<tr>
<td>3</td>
<td>2018/09/24</td>
<td>Mid-Autumn Festival (Day off)</td>
</tr>
<tr>
<td>4</td>
<td>2018/10/01</td>
<td>Data Science and Big Data Analytics: Discovering, Analyzing, Visualizing and Presenting Data</td>
</tr>
<tr>
<td>5</td>
<td>2018/10/08</td>
<td>Fundamental Big Data: MapReduce Paradigm, Hadoop and Spark Ecosystem</td>
</tr>
<tr>
<td>6</td>
<td>2018/10/15</td>
<td>Foundations of Big Data Mining in Python</td>
</tr>
<tr>
<td>7</td>
<td>2018/10/22</td>
<td>Supervised Learning: Classification and Prediction</td>
</tr>
<tr>
<td>8</td>
<td>2018/10/29</td>
<td>Unsupervised Learning: Cluster Analysis</td>
</tr>
<tr>
<td>9</td>
<td>2018/11/05</td>
<td>Unsupervised Learning: Association Analysis</td>
</tr>
<tr>
<td>Week</td>
<td>Date</td>
<td>Subject/Topics</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>10</td>
<td>2018/11/12</td>
<td>Midterm Project Report</td>
</tr>
<tr>
<td>11</td>
<td>2018/11/19</td>
<td>Machine Learning with Scikit-Learn in Python</td>
</tr>
<tr>
<td>12</td>
<td>2018/11/26</td>
<td>Deep Learning for Finance Big Data with TensorFlow</td>
</tr>
<tr>
<td>13</td>
<td>2018/12/03</td>
<td>Convolutional Neural Networks (CNN)</td>
</tr>
<tr>
<td>14</td>
<td>2018/12/10</td>
<td>Recurrent Neural Networks (RNN)</td>
</tr>
<tr>
<td>15</td>
<td>2018/12/17</td>
<td>Reinforcement Learning (RL)</td>
</tr>
<tr>
<td>16</td>
<td>2018/12/24</td>
<td>Social Network Analysis (SNA)</td>
</tr>
<tr>
<td>17</td>
<td>2018/12/31</td>
<td>Bridge Holiday (Extra Day Off)</td>
</tr>
<tr>
<td>18</td>
<td>2019/01/07</td>
<td>Final Project Presentation</td>
</tr>
</tbody>
</table>
Convolutional Neural Networks (CNN)
Outline

• Convolutional Neural Networks (CNN)

• TensorFlow Image Recognition
Artificial Intelligence (AI)

Machine Learning (ML)

Supervised Learning

Unsupervised Learning

Deep Learning (DL)

CNN
RNN LSTM GRU
GAN

Semi-supervised Learning

Reinforcement Learning

Source: https://leonardoaraujosantos.gitbooks.io/artificial-intelligence/content/deep_learning.html
Convolutional Neural Networks (CNN)

Architecture of LeNet-5 (7 Layers) (LeCun et al., 1998)


Convolutional Neural Networks (CNN)

- Convolution
- Pooling
- Fully Connection (FC) (Flattening)
A friendly introduction to Convolutional Neural Networks and Image Recognition

Convolution Layer

Pooling Layer

Source: Luis Serrano (2017), A friendly introduction to Convolutional Neural Networks and Image Recognition, https://www.youtube.com/watch?v=2-OI7ZB0MmU
A friendly introduction to Convolutional Neural Networks and Image Recognition

Source: Luis Serrano (2017), A friendly introduction to Convolutional Neural Networks and Image Recognition, https://www.youtube.com/watch?v=2-OlZB0MmU
A friendly introduction to Convolutional Neural Networks and Image Recognition

Source: Luis Serrano (2017), A friendly introduction to Convolutional Neural Networks and Image Recognition, https://www.youtube.com/watch?v=2-OI7ZB0MmU
A friendly introduction to Convolutional Neural Networks and Image Recognition

Source: Luis Serrano (2017), A friendly introduction to Convolutional Neural Networks and Image Recognition, https://www.youtube.com/watch?v=2-OJ7ZB0MmU
CNN Architecture

**CNN Convolution Layer**

Convolution is a **mathematical operation** to merge two sets of information.

### 3x3 convolution

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

**Input**

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Filter / Kernel**

CNN Convolution Layer

Input x Filter --> Feature Map

receptive field: 3x3

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1x1</td>
<td>1x0</td>
<td>1x1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x0</td>
<td>1x1</td>
<td>1x0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0x1</td>
<td>0x0</td>
<td>1x1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Input x Filter

Feature Map

### CNN Convolution Layer

**Input x Filter --> Feature Map**

**receptive field: 3x3**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>1x1</th>
<th>1x0</th>
<th>0x1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Input x Filter**

**Feature Map**

---

Example convolution operation shown in 2D using a 3x3 filter
CNN Convolution Layer

10 different filters 10 feature maps of size 32x32x1

final output of the convolution layer: a volume of size 32x32x10

CNN Convolution Layer

Sliding operation at 4 locations

CNN Convolution Layer

two feature maps
**CNN Convolution Layer**

*Stride* specifies how much we move the convolution filter at each step.

**Stride 1**

**Feature Map**
**CNN Convolution Layer**

**Stride** specifies how much we move the convolution filter at each step.

![Feature Map](image)

**Stride 2**

CNN Convolution Layer

Stride 1 with Padding

Feature Map

## CNN Pooling Layer

### Max Pooling

Max pooling is a technique used in convolutional neural networks to reduce the dimensionality of the feature maps. It involves taking the maximum value within a defined window (also known as the pooling window) and assigning that value to the output feature map. The window size, often denoted as $2 \times 2$, refers to the size of the area over which the maximum is calculated, and the stride, typically 2, indicates the number of steps taken over the input volume.

Consider a 2D input tensor with dimensions 4x4. When applying max pooling with a $2 \times 2$ window and a stride of 2, the process involves dividing the input into overlapping $2 \times 2$ regions and selecting the maximum value from each.

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

The max pooling operation works as follows:

- **Input:**
  - Two-dimensional data with dimensions 4x4.
  - The values are: 1, 1, 2, 4, 5, 6, 7, 8, 3, 2, 1, 0, 1, 2, 3, 4.

- **Window Size:** $2 \times 2$.
- **Stride:** 2.

- **Output:**
  - The output is a 2x2 matrix with the maximum values from each $2 \times 2$ region of the input.
  - The output values are: 6, 8, 3, 4.

### Illustration

- **Input Matrix:**
  - The values are arranged in a 4x4 grid.

- **Pooling Window:**
  - The $2 \times 2$ window slides over the input matrix.

- **Output Matrix:**
  - The output is reduced to a 2x2 grid, showing the maximum values in each $2 \times 2$ region.

### Source

CNN Pooling Layer

CNN Architecture
4 convolution + pooling layers, followed by 2 fully connected layers

CNN Architecture
4 convolution + pooling layers, followed by 2 fully connected layers

model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', padding='same', name='conv_1', input_shape=(150, 150, 3)))
model.add(MaxPooling2D((2, 2), name='maxpool_1'))
model.add(Conv2D(64, (3, 3), activation='relu', padding='same', name='conv_2'))
model.add(MaxPooling2D((2, 2), name='maxpool_2'))
model.add(Conv2D(128, (3, 3), activation='relu', padding='same', name='conv_3'))
model.add(MaxPooling2D((2, 2), name='maxpool_3'))
model.add(Conv2D(128, (3, 3), activation='relu', padding='same', name='conv_4'))
model.add(MaxPooling2D((2, 2), name='maxpool_4'))
model.add(Flatten())
model.add(Dropout(0.5))
model.add(Dense(512, activation='relu', name='dense_1'))
model.add(Dense(128, activation='relu', name='dense_2'))
model.add(Dense(1, activation='sigmoid', name='output'))

Dropout

No Dropout

With Dropout

Model Performance

Train Loss: 0.054, Val Loss: 1.345

Train Accuracy: 0.981, Val Accuracy: 0.732

Source: Arden Dertat (2017), Applied Deep Learning - Part 4: Convolutional Neural Networks,
https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2
Visual Recognition

Image Classification
Is this a Cat or Dog?

Source: Jeff Dean (2016), Large-Scale Deep Learning For Building Intelligent Computer Systems, WSDM 2016
Convolutional Neural Networks (CNNs / ConvNets)

http://cs231n.github.io/convolutional-networks/
A regular 3-layer Neural Network

http://cs231n.github.io/convolutional-networks/
A ConvNet arranges its neurons in three dimensions (width, height, depth)

http://cs231n.github.io/convolutional-networks/
The activations of an example ConvNet architecture.

http://cs231n.github.io/convolutional-networks/
ConvNets

32x32x3 CIFAR-10 image

first Convolutional layer

http://cs231n.github.io/convolutional-networks/
ConvNets

http://cs231n.github.io/convolutional-networks/
### Convolution Demo

<table>
<thead>
<tr>
<th>Input Volume (+pad 1) (7x7x3)</th>
<th>Filter W0 (3x3x3)</th>
<th>Filter W1 (3x3x3)</th>
<th>Output Volume (3x3x2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x[:, :, 0])</td>
<td>(w0[:, :, 0])</td>
<td>(w1[:, :, 0])</td>
<td>(o[:, :, 0])</td>
</tr>
<tr>
<td>0 0 0 0 0 0 0</td>
<td>-1 -1 0</td>
<td>1 -1 0</td>
<td>6 3 6</td>
</tr>
<tr>
<td>0 1 2 0 2 1 0</td>
<td>1 1 1</td>
<td>0 1 1</td>
<td>7 -1 -2</td>
</tr>
<tr>
<td>0 2 2 2 1 1 0</td>
<td>-1 0 1</td>
<td>0 -1 1</td>
<td>2 3 -2</td>
</tr>
<tr>
<td>0 2 2 2 0 1 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 2 1 2 1 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 1 2 0 1 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(x[:, :, 1])</td>
<td>(w0[:, :, 1])</td>
<td>(w1[:, :, 1])</td>
<td>(o[:, :, 1])</td>
</tr>
<tr>
<td>0 0 0 0 0 0 0</td>
<td>0 0 1</td>
<td>-1 1 0</td>
<td>4 3 2</td>
</tr>
<tr>
<td>0 0 2 2 1 2 0</td>
<td>0 1 0</td>
<td>-1 -1 1</td>
<td>-1 0 -1</td>
</tr>
<tr>
<td>0 1 2 0 0 2 0</td>
<td>-1 0 -1</td>
<td>0 0 0</td>
<td></td>
</tr>
<tr>
<td>0 0 1 2 1 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 2 2 2 2 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 2 2 2 0 2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(x[:, :, 2])</td>
<td>(w0[:, :, 2])</td>
<td>(w1[:, :, 2])</td>
<td>(\text{Bias } b0 (1x1x1))</td>
</tr>
<tr>
<td>0 0 0 0 0 0 0</td>
<td>-1 -1 0</td>
<td>1 0 -1</td>
<td>(b0[:, :, 0])</td>
</tr>
<tr>
<td>0 1 0 0 1 0</td>
<td>1 0 -1</td>
<td>0 0 -1</td>
<td>0</td>
</tr>
<tr>
<td>0 0 2 0 0 0</td>
<td>-1 0 -1</td>
<td>1 0 1</td>
<td></td>
</tr>
<tr>
<td>0 0 0 1 1 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 2 2 2 1 2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 1 2 0 2 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(\text{Bias } b1 (1x1x1))</td>
<td>(b1[:, :, 0])</td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

ConvNets

input volume of size [224x224x64] is pooled with filter size 2, stride 2 into output volume of size [112x112x64]

http://cs231n.github.io/convolutional-networks/
ConvNets
max pooling

http://cs231n.github.io/convolutional-networks/
Convolutional Neural Networks (CNN) (LeNet)

Source: [http://deeplearning.net/tutorial/lenet.html](http://deeplearning.net/tutorial/lenet.html)
Convolutional Neural Networks (CNN) (LeNet)

example of a convolutional layer

Source: http://deeplearning.net/tutorial/lenet.html
Neural Network and Deep Learning

Source: 3Blue1Brown (2017), But what *is* a Neural Network? | Chapter 1, deep learning, https://www.youtube.com/watch?v=aircAruvnKk
Gradient Descent
how neural networks learn

Average cost of all training data...

\[
\begin{align*}
&= (0.18 - 0.00)^2 + \\
&= (0.29 - 0.00)^2 + \\
&= (0.58 - 0.00)^2 + \\
&= (0.77 - 0.00)^2 + \\
&= (0.20 - 0.00)^2 + \\
&= (0.36 - 0.00)^2 + \\
&= (0.93 - 0.00)^2 + \\
&= (1.00 - 0.00)^2 + \\
&= (0.95 - 1.00)^2 + \\
&= (0.35 - 0.00)^2
\end{align*}
\]

What’s the “cost” of this difference?

Source: 3Blue1Brown (2017), Gradient descent, how neural networks learn | Chapter 2, deep learning,
https://www.youtube.com/watch?v=IHZwWFHWa-w
Backpropagation

Source: 3Blue1Brown (2017), What is backpropagation really doing? | Chapter 3, deep learning,
https://www.youtube.com/watch?v=Ilg3gGewQ5U
TensorFlow
Image
Recognition
MNIST dataset:
60,000 labeled digits

Training digits

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#0
Get Started with TensorFlow

TensorFlow is an open-source machine learning library for research and production. TensorFlow offers APIs for beginners and experts to develop for desktop, mobile, web, and cloud. See the sections below to get started.

Learn and use ML

The high-level Keras API provides building blocks to create and train deep learning models. Start with these beginner-friendly notebook examples, then read the TensorFlow Keras guide.

1. Basic classification
2. Text classification
3. Regression
4. Overfitting and underfitting
5. Save and load

```python
import tensorflow as tf
mnist = tf.keras.datasets.mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()
(x_train, x_test) = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([tf.keras.layers.Flatten(),
                       tf.keras.layers.Dense(512, activation=tf.nn.relu),
                       tf.keras.layers.Dropout(0.2),
                       tf.keras.layers.Dense(10, activation=tf.nn.softmax)])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```

https://www.tensorflow.org/tutorials/
import tensorflow as tf

import tensorflow as tf

mnist = tf.keras.datasets.mnist

(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([tf.keras.layers.Flatten(),tf.keras.layers.Dense(512, activation=tf.nn.relu),tf.keras.layers.Dropout(0.2),tf.keras.layers.Dense(10, activation=tf.nn.softmax)])

model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)

model.evaluate(x_test, y_test)

https://www.tensorflow.org/tutorials/
Get Started with TensorFlow

This is a Google Colaboratory notebook file. Python programs are run directly in the browser—a great way to learn and use TensorFlow. To run the Colab notebook:

1. Connect to a Python runtime: At the top-right of the menu bar, select CONNECT.
2. Run all the notebook code cells: Select Runtime > Run all.

For more examples and guides (including details for this program), see Get Started with TensorFlow.

Let's get started, import the TensorFlow library into your program:

```python
import tensorflow as tf
```

Load and prepare the MNIST dataset. Convert the samples from integers to floating-point numbers:

```python
mnist = tf.keras.datasets.mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0
```

Build the tf.keras model by stacking layers. Select an optimizer and loss function used for training:

```python
model = tf.keras.models.Sequential([tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation=tf.nn.relu),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(10, activation=tf.nn.softmax)]
)
```
1. Overview

In this codelab, you will learn how to build and train a neural network that recognises handwritten digits. Along the way, as you enhance your neural network to achieve 99% accuracy, you will also discover the tools of the trade that deep learning professionals use to train their models efficiently.

This codelab uses the MNIST dataset, a collection of 60,000 labeled digits that has kept generations of PhDs busy for almost two decades. You will solve the problem with less than 100 lines of Python / TensorFlow code.
Sample code for "Tensorflow and deep learning, without a PhD" presentation and code lab.

Commit: "miengine" added example using the Tensorflow high level layers API
Commit: "CONTRIBUTING.md" initial commit
Commit: "LICENSE" initial commit
Commit: "README.md" better image URL
Commit: "mnist_1.0_softmax.py" global_variables_initializer used everywhere instead of initialize_al...
Commit: "mnist_2.0_five_layers_sigmoid.py" Fix spacing in the network structure comment
Commit: "mnist_2.1_five_layers_relu_lrdecay..." Fix spacing in the network structure comment

https://github.com/martin-gorner/tensorflow-mnist-tutorial/
TensorFlow and Deep Learning

• What is a neural network and how to train it
• How to build a basic 1-layer neural network using TensorFlow
• How to add more layers
• Training tips and tricks: overfitting, dropout, learning rate decay ...
• How to troubleshoot deep neural networks
• How to build convolutional networks

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#0
TensorFlow MNIST Tutorial

```
git clone https://github.com/martin-gorner/tensorflow-mnist-tutorial.git

cd tensorflow-mnist-tutorial

python3 mnist_1.0_softmax.py

python mnist_1.0_softmax.py

pythonw mnist_1.0_softmax.py
```

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#1
Train a Neural Network

Training digits
updates to weights and biases =>
better recognition (loop)

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#2
Training digits

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#2
Test digits

Accuracy

Cross entropy loss

Training digits

Weights

Biases

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#2
Cross entropy loss

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#2
Accuracy

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#2
Biases

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#2
Weights and Biases

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#2
Cookbook

Softmax
Cross-entropy
Mini-batch
Very Simple Model: Softmax Classification

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/
Very Simple Model: Softmax Classification

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/
Very Simple Model: Softmax Classification

\[ \text{softmax}(L_n) = \frac{e^{L_n}}{\|e^L\|} \]

weighted sum of all pixels + bias

neuron outputs

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/
In Matrix notation, 100 images at a time

X: 100 images, one per line, flattened

\[
\begin{bmatrix}
W_{0,0} & W_{0,1} & W_{0,2} & W_{0,3} & \cdots & W_{0,9} \\
W_{1,0} & W_{1,1} & W_{1,2} & W_{1,3} & \cdots & W_{1,9} \\
W_{2,0} & W_{2,1} & W_{2,2} & W_{2,3} & \cdots & W_{2,9} \\
W_{3,0} & W_{3,1} & W_{3,2} & W_{3,3} & \cdots & W_{3,9} \\
W_{4,0} & W_{4,1} & W_{4,2} & W_{4,3} & \cdots & W_{4,9} \\
W_{5,0} & W_{5,1} & W_{5,2} & W_{5,3} & \cdots & W_{5,9} \\
W_{6,0} & W_{6,1} & W_{6,2} & W_{6,3} & \cdots & W_{6,9} \\
W_{7,0} & W_{7,1} & W_{7,2} & W_{7,3} & \cdots & W_{7,9} \\
W_{8,0} & W_{8,1} & W_{8,2} & W_{8,3} & \cdots & W_{8,9} \\
\cdots & & & & & \\
W_{783,0} & W_{783,1} & W_{783,2} & \cdots & W_{783,9}
\end{bmatrix}
\]

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/
X: 100 images, one per line, flattened

\[
\begin{bmatrix}
W_{0,0} & W_{0,1} & W_{0,2} & W_{0,3} & \cdots & W_{0,9} \\
W_{1,0} & W_{1,1} & W_{1,2} & W_{1,3} & \cdots & W_{1,9} \\
W_{2,0} & W_{2,1} & W_{2,2} & W_{2,3} & \cdots & W_{2,9} \\
W_{3,0} & W_{3,1} & W_{3,2} & W_{3,3} & \cdots & W_{3,9} \\
W_{4,0} & W_{4,1} & W_{4,2} & W_{4,3} & \cdots & W_{4,9} \\
W_{5,0} & W_{5,1} & W_{5,2} & W_{5,3} & \cdots & W_{5,9} \\
W_{6,0} & W_{6,1} & W_{6,2} & W_{6,3} & \cdots & W_{6,9} \\
W_{7,0} & W_{7,1} & W_{7,2} & W_{7,3} & \cdots & W_{7,9} \\
W_{8,0} & W_{8,1} & W_{8,2} & W_{8,3} & \cdots & W_{8,9} \\
\vdots \\
W_{783,0} & W_{783,1} & W_{783,2} & \cdots & W_{783,9}
\end{bmatrix}
\]

784 pixels
X: 100 images, one per line, flattened

\[ W_{0,0} \quad W_{0,1} \quad W_{0,2} \quad W_{0,3} \quad \ldots \quad W_{0,9} \]
\[ W_{1,0} \quad W_{1,1} \quad W_{1,2} \quad W_{1,3} \quad \ldots \quad W_{1,9} \]
\[ W_{2,0} \quad W_{2,1} \quad W_{2,2} \quad W_{2,3} \quad \ldots \quad W_{2,9} \]
\[ W_{3,0} \quad W_{3,1} \quad W_{3,2} \quad W_{3,3} \quad \ldots \quad W_{3,9} \]
\[ W_{4,0} \quad W_{4,1} \quad W_{4,2} \quad W_{4,3} \quad \ldots \quad W_{4,9} \]
\[ W_{5,0} \quad W_{5,1} \quad W_{5,2} \quad W_{5,3} \quad \ldots \quad W_{5,9} \]
\[ W_{6,0} \quad W_{6,1} \quad W_{6,2} \quad W_{6,3} \quad \ldots \quad W_{6,9} \]
\[ W_{7,0} \quad W_{7,1} \quad W_{7,2} \quad W_{7,3} \quad \ldots \quad W_{7,9} \]
\[ W_{8,0} \quad W_{8,1} \quad W_{8,2} \quad W_{8,3} \quad \ldots \quad W_{8,9} \]
\[ \ldots \]
\[ W_{783,0} \quad W_{783,1} \quad W_{783,2} \quad W_{783,3} \quad \ldots \quad W_{783,9} \]

8x4x64x3

784 pixels

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/
### 10 columns

<table>
<thead>
<tr>
<th>$W_{0,0}$</th>
<th>$W_{0,1}$</th>
<th>$W_{0,2}$</th>
<th>$W_{0,3}$</th>
<th>...</th>
<th>$W_{0,9}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$W_{1,0}$</td>
<td>$W_{1,1}$</td>
<td>$W_{1,2}$</td>
<td>$W_{1,3}$</td>
<td>...</td>
<td>$W_{1,9}$</td>
</tr>
<tr>
<td>$W_{2,0}$</td>
<td>$W_{2,1}$</td>
<td>$W_{2,2}$</td>
<td>$W_{2,3}$</td>
<td>...</td>
<td>$W_{2,9}$</td>
</tr>
<tr>
<td>$W_{3,0}$</td>
<td>$W_{3,1}$</td>
<td>$W_{3,2}$</td>
<td>$W_{3,3}$</td>
<td>...</td>
<td>$W_{3,9}$</td>
</tr>
<tr>
<td>$W_{4,0}$</td>
<td>$W_{4,1}$</td>
<td>$W_{4,2}$</td>
<td>$W_{4,3}$</td>
<td>...</td>
<td>$W_{4,9}$</td>
</tr>
<tr>
<td>$W_{5,0}$</td>
<td>$W_{5,1}$</td>
<td>$W_{5,2}$</td>
<td>$W_{5,3}$</td>
<td>...</td>
<td>$W_{5,9}$</td>
</tr>
<tr>
<td>$W_{6,0}$</td>
<td>$W_{6,1}$</td>
<td>$W_{6,2}$</td>
<td>$W_{6,3}$</td>
<td>...</td>
<td>$W_{6,9}$</td>
</tr>
<tr>
<td>$W_{7,0}$</td>
<td>$W_{7,1}$</td>
<td>$W_{7,2}$</td>
<td>$W_{7,3}$</td>
<td>...</td>
<td>$W_{7,9}$</td>
</tr>
<tr>
<td>$W_{8,0}$</td>
<td>$W_{8,1}$</td>
<td>$W_{8,2}$</td>
<td>$W_{8,3}$</td>
<td>...</td>
<td>$W_{8,9}$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
<td>...</td>
</tr>
<tr>
<td>$W_{783,0}$</td>
<td>$W_{783,1}$</td>
<td>$W_{783,2}$</td>
<td>...</td>
<td>$W_{783,9}$</td>
<td></td>
</tr>
</tbody>
</table>

#### 784 Lines

**X:** 100 images, one per line, flattened

**784 pixels**

Source: [https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/](https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/)
X: 100 images, one per line, flattened
What are "weights" and "biases"?

How is the "cross-entropy" computed?

How exactly does the training algorithm work?
\[ Y = f(X) \]

- Predictions: \( Y[100, 10] \)
- Images: \( X[100, 784] \)
- Weights: \( W[784, 10] \)
- Biases: \( b[10] \)

\[
Y = \text{softmax}(X \cdot W + b)
\]

- Applied line by line
- Matrix multiply
- Broadcast on all lines

\[
Y = \text{tf.nn.softmax(tf.matmul}(X, W) + b)
\]
TensorFlow (Python) Softmax

\[ Y = \text{tf.nn.softmax}(\text{tf.matmul}(X, W) + b) \]

Predictions:
\[ Y[100, 10] \]

tensor shapes: \[ X[100, 784] \quad W[784, 10] \quad b[10] \]

matrix multiply
broadcast on all lines

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/
Cross Entropy:

$$- \sum Y'_i \cdot \log(Y_i)$$

actual probabilities, "one-hot" encoded

computed probabilities

this is a "6"
Minimizing Cross Entropy (Minimizing Loss)

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/
Training Loop

Training digits and labels
=> loss function
=> gradient (partial derivatives)
=> steepest descent
=> update weights and biases
=> repeat with next mini-batch of training images and labels

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/
"mini-batches":
100 images and labels
import tensorflow as tf
import tensorflow as tf
X = tf.placeholder(tf.float32, [None, 28, 28, 1])
W = tf.Variable(tf.zeros([784, 10]))
b = tf.Variable(tf.zeros([10]))

init = tf.initialize_all_variables()
# model
Y = tf.nn.softmax(tf.matmul(tf.reshape(X, [-1, 784]), W) + b)

# placeholder for correct labels
Y_ = tf.placeholder(tf.float32, [None, 10])

# loss function
cross_entropy = -tf.reduce_sum(Y_ * tf.log(Y))

# % of correct answers found in batch
is_correct = tf.equal(tf.argmax(Y, 1), tf.argmax(Y_, 1))
accuracy = tf.reduce_mean(tf.cast(is_correct, tf.float32))

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#5
sess = tf.Session()
sess.run(init)

for i in range(1000):
    # load batch of images and correct answers
    batch_X, batch_Y = mnist.train.next_batch(100)
    train_data={X: batch_X, Y_: batch_Y}

    # train
    sess.run(train_step, feed_dict=train_data)
# success ?
a,c = sess.run([accuracy, cross_entropy],
feed_dict=train_data)

# success on test data ?
test_data={X: mnist.test.images, Y_: mnist.test.labels}
a,c = sess.run([accuracy, cross_entropy], feed=test_data)
mnist_1.0_softmax.py

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#5
Deep Learning

Go deep!
5 fully-connected layers

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/
Sigmoid

\[
\frac{1}{1 + e^{-x}}
\]

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/
5 fully-connected layers

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/
TensorFlow MNIST Tutorial

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/
ReLU

Relu!
TensorFlow MNIST Tutorial
ReLU
TensorFlow MNIST Tutorial
Learning Rate

Slow down…

Learning rate decay

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/
LR = 0.003

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/
TensorFlow MNIST Tutorial

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/
Dropout

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/
Overfitting

Accuracy

Cross entropy loss

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/
Dropout

Training
pkeep = 0.75

Test
pkeep = 1.0
TensorFlow MNIST Tutorial

Accuracy

Cross entropy loss

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/
TensorFlow MNIST Tutorial

Accuracy

Cross entropy loss

5 layers Sigmoid

97.9%

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/
Overfitting

Overfitting ?!? Too many neurons

BAD Network

Not enough DATA

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/
Convolutional Layer

\[ W_{[4, 4, 3]} \]

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/
Convolutional Layer

\[ W_{[4, 4, 3]} \]

\[ W_{2}^{[4, 4, 3]} \]

filter size

input channels

output channels

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/
Convolutional Max-Pool

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/
All Convolutional

Hacker's tip

ALL Convolutional

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/
Bigger Convolutional Neural Network

convolutional layer, 4 channels
\( W_1[5, 5, 1, 4] \) stride 1

convolutional layer, 8 channels
\( W_2[4, 4, 4, 8] \) stride 2

convolutional layer, 12 channels
\( W_3[4, 4, 8, 12] \) stride 2

fully connected layer
\( W_4[7 \times 7 \times 12, 200] \)

softmax readout layer
\( W_5[200, 10] \)

+ biases on all layers

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/
Bigger Convolutional Neural Network

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/
Bigger Convolutional Neural Network + Dropout

convolutional layer, 6 channels
$W_1[6, 6, 1, 6]$ stride 1

convolutional layer, 12 channels
$W_2[5, 5, 6, 12]$ stride 2

convolutional layer, 24 channels
$W_3[4, 4, 12, 24]$ stride 2

fully connected layer
$W_4[7x7x24, 200]$ softmax readout layer
$W_5[200, 10]$

+ biases on all layers

28x28x1
28x28x6
14x14x12
7x7x24
200
10

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/
TensorFlow MNIST Tutorial

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/
 TensorFlow MNIST Tutorial

Accuracy

Cross entropy loss

larger convolutional network

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/
TensorFlow MNIST Tutorial

Accuracy

Cross entropy loss

Max activations across batch

Training digits

Test digits

Source: https://github.com/martin-gorner/tensorflow-mnist-tutorial/
TensorFlow MNIST Tutorial

Source: https://github.com/martin-gorner/tensorflow-mnist-tutorial/
Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/
AI + VDI POS
TensorFlow Models

• M1: Basic Classification (Image Classification) (65 Seconds)

• M2: Basic Text Classification (Text Classification) (46 Seconds)

• M3: Basic Regression (Predict House Prices) (43 Seconds)

• M4: Pix2Pix Eager (Option) (7-8 Hours)

• M5: NMT with Attention (Option) (20-30 minutes)
Basic Classification

Fashion MNIST Image Classification

https://colab.research.google.com/drive/19PJOJi1vn1kjculzNHjRSLbeVl4kd5z

Train your first neural network: basic classification

This guide trains a neural network model to classify images of clothing, like sneakers and shirts. It's okay if you don't understand all the details, this is a fast-paced overview of a complete TensorFlow program with the details explained as we go.

This guide uses TensorFlow, a high-level API to build and train models in TensorFlow.
Text Classification
IMDB Movie Reviews

https://colab.research.google.com/drive/1x16h1GhHsL1r1ytPCvCHaoO1W-i_gror

Text classification with movie reviews

Download the IMDB dataset

Explore the data

Convert the integers back to words

Prepare the data

Build the model

Hidden units

Loss function and optimizer

Create a validation set

Train the model

Evaluate the model

Copyright 2018 The TensorFlow Authors.

Licensed under the Apache License, Version 2.0 (the "License");
MIT License

This notebook classifies movie reviews as positive or negative using the text of the review. This is an example of binary—or two-class—classification, an important and widely applicable kind of machine learning problem.

We'll use the IMDB dataset that contains the text of 50,000 movie reviews from the Internet Movie Database. These are split into 25,000 reviews for training and 25,000 reviews for testing. The training and testing sets are balanced, meaning they contain an equal number of positive and negative reviews.

This notebook uses tf.keras, a high-level API to build and train models in TensorFlow. For a more advanced text classification tutorial using tf.keras, see the MLCC Text Classification Guide.

```python
# memory footprint support libraries/code
!
ln -s /opt/bin/nvidia-smi /usr/bin/nvidia-smi
!pip install gputil
!pip install putil
!pip install humanize
import putil
import humanize
import os
import GPUTil as GPU
GPUs = GPU.getGPUs()
gpu = GPUs[0]
def print():
```
Basic Regression

Predict House Prices

https://colab.research.google.com/drive/1v4c8ZHtnRtg2_25K_AURj6SCVBRdlj

Copyright 2018 The TensorFlow Authors.

Predict house prices: regression

The Boston Housing Prices dataset

Examples and features

Labels

Normalize features

Create the model

Train the model

Predict

Conclusion

---

Copyright 2018 The TensorFlow Authors.

2 cells hidden

Predict house prices: regression

In a regression problem, we aim to predict the output of a continuous value, like a price or a probability. Contrast this with a classification problem, where we aim to predict a discrete label (for example, where a picture contains an apple or an orange).

This notebook builds a model to predict the median price of homes in a Boston suburb during the mid-1970s. To do this, we'll provide the model with some data points about the suburb, such as the crime rate and the local property tax rate.

This example uses the tf.keras API, see this guide for details.

```python
# memory footprint support libraries/code
!ln -sf /opt/bin/nvidia-smi /usr/bin/nvidia-smi
!pip install gputil
!pip install psutil
!pip install humanize
import psutil
import humanize
import os
import GPUtil as GPU
GPUs = GPU.getGPUs()
gpu = GPUs[0]
def printm():
    process = psutil.Process(gpu.pid)
    print("GPU RAM Free: " + humanize.naturalsize( psutil.virtual_memory().available ), " | Proc size:")
    print("GPU RAM Free: {0:.0f}MB | Used: {1:.0f}MB | Util {2:.0f}% | Total {3:.0f}MB".format(gpu.memory_usage[0]))
```
AI+VDI POC
ISAC+TKU Test

• AI+VDI POC Folder (3+1 ipynb) (v3.0.20181120)
  – https://drive.google.com/open?id=1qHOemktbEmUz-ot8eFxIKbGwJvXlrjtc

• run3models.ipynb
  – https://colab.research.google.com/drive/1HQ1GrIqQUUPCct7_AVgoMwMrh0UqMm0f
Summary

• Convolutional Neural Networks (CNN)

• TensorFlow Image Recognition
References

- Luis Serrano (2017), A friendly introduction to Convolutional Neural Networks and Image Recognition, https://www.youtube.com/watch?v=2-Ol7ZB0MmU
- Martin Gorner (2017), TensorFlow and Deep Learning without a PhD, Part 2 (Google Cloud Next '17), https://www.youtube.com/watch?v=fTUwdXUFl8
- Deep Learning Basics: Neural Networks Demystified, https://www.youtube.com/playlist?list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU
- Deep Learning SIMPLIFIED, https://www.youtube.com/playlist?list=PLjJh1vlSEYgvGod9wWiydumYl8hOXixNu
- 3Blue1Brown (2017), But what *is* a Neural Network? | Chapter 1, deep learning, https://www.youtube.com/watch?v=aircAruvnKk
- 3Blue1Brown (2017), Gradient descent, how neural networks learn | Chapter 2, deep learning, https://www.youtube.com/watch?v=IHZwWFHWa-w
- 3Blue1Brown (2017), What is backpropagation really doing? | Chapter 3, deep learning, https://www.youtube.com/watch?v=Ilg3gGewQ5U
- TensorFlow: https://www.tensorflow.org/
- Keras: http://keras.io/
- Udacity, Deep Learning, https://www.youtube.com/playlist?list=PLAwxTw4SYaPn_OWPFT9ulXLuQrlmzHfOV
- https://github.com/leriomaggio/deep-learning-keras-tensorflow