TensorFlow
symbolic computing for machine learning

this is about creating models

X
observations

MODEL

Y
predictions

raul ramos - UdeA
ANALYTIC MODELS

\[ \frac{1}{2}Mv^2 + \frac{1}{2}mv^2 - Mgh + mg\sin\theta = 0 \]

\[ \frac{1}{2}(m + M)v^2 = gh(M - m\sin\theta) \]

\[ v = \sqrt{\frac{2gh(M - m\sin\theta)}{m + M}} \]
MACHINE LEARNING

DATA → ML → X → MODEL → Y
observations → predictions
SUPERVISED LEARNING

**X**
- Patient data
- Images
- Financial data
- Robot sensors
- Flight history
- Movies watched
- Recorded voice
- Driving data

**Y**
- Presence of pathology
- Objects present
- Next price
- Steering action
- Landing time
- Movies to suggest
- Words transcribed
- Driver profile

**MODEL TRAINING**

**MODEL**

show video
FAMILIES of MODELS

DATA

ADJUSTED MODEL

Decision tree

Neural network

3 degree SVM
Tenemos un dataset de entrenamiento \( \{(x^{(i)}, y^{(i)})\} \) con \( i \in \{1...m\} \) (es decir, con \( m \) puntos), \( x^{(i)} = [1 \ x^{(i)}_1 \ x^{(i)}_2 \ ... \ x^{(i)}_j \ ... \ x^{(i)}_n]^T \in \mathbb{R}^{n+1} \) y \( y^{(i)} \in \{0, 1\} \) para clasificación binaria. La matriz \( \mathbf{X} \) recoge todos los \( x^{(i)} \) y el vector \( \mathbf{y} \) todos los \( y^{(i)} \)

\[
\mathbf{X} = \begin{bmatrix}
\_ \_ \_ \_ \_ \_ x^{(1)} & \_ \_ \_ \_ \_ \\
\_ \_ \_ \_ \_ \_ x^{(2)} & \_ \_ \_ \_ \\
\_ \_ \_ \_ \_ \_ \_ \_ x^{(k)} & \_ \_ \_ \_ \\
\_ \_ \_ \_ \_ \_ \_ \_ \_ x^{(m)} & \_ \_ \\
\end{bmatrix}
= \begin{bmatrix}
1 \ x^{(1)}_1 \ x^{(1)}_2 \ ... \ x^{(1)}_n \\
1 \ x^{(2)}_1 \ x^{(2)}_2 \ ... \ x^{(2)}_n \\
1 \ x^{(k)}_1 \ x^{(k)}_2 \ ... \ x^{(k)}_n \\
1 \ x^{(m)}_1 \ x^{(m)}_2 \ ... \ x^{(m)}_n \\
\end{bmatrix}
\quad \mathbf{y} = \begin{bmatrix}
y^{(1)} \\
y^{(2)} \\
\vdots \\
y^{(m)} \\
\end{bmatrix}
\]

\( \hat{y}^{(i)} = f(x^{(i)}) \) where \( f \) is our model

\( m: \) número de datos \quad \( n: \) número de descriptores por dato
Tenemos un dataset de entrenamiento \( \{(x^{(i)}, y^{(i)})\} \) con \( i \in \{1...m\} \) (es decir, con \( m \) puntos), \( x^{(i)} = [1 \ x_1^{(i)} \ x_2^{(i)} \ ... \ x_j^{(i)} \ ... \ x_n^{(i)}]^T \in \mathbb{R}^{n+1} \) y \( y^{(i)} \in \{0, 1\} \) para clasificación binaria. La matriz \( \mathbf{X} \) recoge todos los \( x^{(i)} \) y el vector \( \mathbf{y} \) todos los \( y^{(i)} \).

\[
\mathbf{X} = \begin{bmatrix}
\vdots & x^{(1)} & \vdots \\
\vdots & x^{(2)} & \vdots \\
\vdots & \ddots & \vdots \\
\vdots & x^{(m)} & \vdots 
\end{bmatrix}
= \begin{bmatrix}
1 & x_1^{(1)} & x_2^{(1)} & \cdots & x_n^{(1)} \\
1 & x_1^{(2)} & x_2^{(2)} & \cdots & x_n^{(2)} \\
\vdots & \ddots & \vdots & \ddots & \vdots \\
1 & x_1^{(m)} & x_2^{(m)} & \cdots & x_n^{(m)}
\end{bmatrix}
\quad \mathbf{y} = \begin{bmatrix}
y^{(1)} \\
y^{(2)} \\
\vdots \\
y^{(m)}
\end{bmatrix}
\]

\( y^{(i)} \in \mathbb{R} \quad \rightarrow \quad \text{Regression} \)

\( y^{(i)} \in [0, 1] \text{ or } [-1, 1] \quad \rightarrow \quad \text{Classification} \)
LINEAR REGRESSION

FORMULATE MODEL FAMILY

X/Y linear relation

DESIGN COST FUNCTION

Avg squared error

arg min \( J(\theta) \)

Gradient descent (or generic optimizer)

Closed form

EVALUATE

Train/test split
Learning curves, Stability of models, etc.

\[ \hat{y}^{(i)} = \theta^T x^{(i)} \]

\[ J(\theta) = \frac{1}{m} \sum_{i=0}^{m-1} (\hat{y}^{(i)} - y^{(i)})^2 \]

\[ \nabla J = \frac{2}{m} X^T \cdot (X \cdot \theta - Y) \]

\[ \theta = (X^T X)^{-1} X^T Y \]
show notebook part 1
PERCEPTRON

FORMULATE MODEL FAMILY

\[ W_1 = \begin{bmatrix} w_0^0 & w_0^1 \\ w_1^0 & w_1^1 \\ w_2^0 & w_2^1 \end{bmatrix} \quad b_1 = [b_0 \quad b_1] \]

\[ \hat{y} = \text{sigmoid} \left( W_2 \cdot \tanh(x^T W_1 + b_1) + b_2 \right) \]
PERCEPTRON

\[
\hat{y} = g(W_2 \cdot \tanh(x^T \cdot W_1 + b_1) + b_2)
\]

\[
J(b_1, b_2, W_1, W_2) = \frac{1}{m} \sum_{i=0}^{m-1} (\hat{y}^{(i)} - y^{(i)})^2.
\]

Gradient descent (or generic optimizer)

BACKPROPAGATION
CONVOLUTIONAL NETWORKS

ml4a.github.io/dev/demos/demo_convolution.html


\[ \hat{y}^{(i)} = \theta^T x^{(i)} \]

\[ J(\theta) = \frac{1}{m} \sum_{i=0}^{m-1} (\hat{y}^{(i)} - y^{(i)})^2 \]

\[ \hat{y} = g(W_2 \cdot \tanh(x^T \cdot W_1 + b_1) + b_2) \]

\[ J(b_1, b_2, W_1, W_2) = \frac{1}{m} \sum_{i=0}^{m-1} (\hat{y}^{(i)} - y^{(i)})^2 . \]
STRUCTURE OF COST FUNCTIONS

\[ \hat{y}^{(i)} = \theta^T x^{(i)} \]

\[ J(\theta) = \frac{1}{m} \sum_{i=0}^{m-1} (\hat{y}^{(i)} - y^{(i)})^2 \]

\[ \hat{y} = g(W_2 \cdot \tanh(x^T \cdot W_1 + b_1) + b_2) \]

\[ J(b_1, b_2, W_1, W_2) = \frac{1}{m} \sum_{i=0}^{m-1} (\hat{y}^{(i)} - y^{(i)})^2 \]
STRUCTURE OF COST FUNCTIONS

\[
\hat{y}^{(i)} = \theta^T x^{(i)}
\]

\[
J(\theta) = \frac{1}{m} \sum_{i=0}^{m-1} (\hat{y}^{(i)} - y^{(i)})^2
\]

\[
\hat{y} = g(W_2 \cdot \tanh(x^T \cdot W_1 + b_1) + b_2)
\]

\[
J(b_1, b_2, W_1, W_2) = \frac{1}{m} \sum_{i=0}^{m-1} (\hat{y}^{(i)} - y^{(i)})^2
\]
STRUCTURE OF COST FUNCTIONS

MATRIX SYMBOLIC COMPUTING
WITH SYMBOLS FOR DATA (no gradients)
AND SYMBOLS FOR PARAMS (need gradients)

\[
\hat{y}(i) = \theta^T x^{(i)}
\]

\[
J(\theta) = \frac{1}{m} \sum_{i=0}^{m-1} (\hat{y}(i) - y^{(i)})^2
\]

\[
\hat{y} = g(W_2 \cdot \tanh(x^T \cdot W_1 + b_1) + b_2)
\]

\[
J(b_1, b_2, W_1, W_2) = \frac{1}{m} \sum_{i=0}^{m-1} (\hat{y}(i) - y^{(i)})^2.
\]
show notebook part 2
CNN

[Diagram of CNN layers with annotations for padding, stride, convolutional subsampling, filter size, input channels, and output channels.]

W[4, 4, 3]
W'[4, 4, 3]
W[4, 4, 3, 2]

C P C P C C C C C C P FC FC FC Label
CNN
“small” CNN

<table>
<thead>
<tr>
<th>layer</th>
<th>input_size</th>
<th>output_size</th>
<th>filter_size</th>
<th>stride</th>
<th>n_filters</th>
<th>activation</th>
<th>W_size from previous</th>
</tr>
</thead>
<tbody>
<tr>
<td>conv1</td>
<td>28x28x1</td>
<td>28x28x9</td>
<td>5x5</td>
<td>1</td>
<td>16</td>
<td>relu</td>
<td>W1 = [5,5,1,16]</td>
</tr>
<tr>
<td>conv2</td>
<td>28x28x16</td>
<td>14x14x8</td>
<td>5x5</td>
<td>2</td>
<td>8</td>
<td>relu</td>
<td>W2 = [5,5,16,8]</td>
</tr>
<tr>
<td>conv3</td>
<td>14x14x8</td>
<td>7x7x12</td>
<td>4x4</td>
<td>2</td>
<td>12</td>
<td>relu</td>
<td>W3 = [4,4,8,12]</td>
</tr>
<tr>
<td>fc</td>
<td>7x7x12</td>
<td>200</td>
<td></td>
<td></td>
<td></td>
<td>relu</td>
<td>W3 = [588,200]</td>
</tr>
<tr>
<td>output</td>
<td>200</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td>softmax</td>
<td>W4 = [200,10]</td>
</tr>
</tbody>
</table>

400 + 3600 + 1536 + 117600 + 2000 = 125136 tunnable params!!!
show notebook Notes 04

show logdir in tensorboard
device placement
show graph in tensorboard from last CNN
device placement

- individual ops have parallel implementations (multi core CPU or multi thread GPU)
- can specify device placement of components of the computation graph (data and/or operations)
- tensorflow will move data around to comply

```python
# Creates a graph.
with tf.device('/device:GPU:2'):
    a = tf.constant([1.0, 2.0, 3.0, 4.0, 5.0, 6.0], shape=[2, 3], name='a')
    b = tf.constant([1.0, 2.0, 3.0, 4.0, 5.0, 6.0], shape=[3, 2], name='b')
    c = tf.matmul(a, b)
    # Creates a session with log_device_placement set to True.
    sess = tf.Session(config=tf.ConfigProto(log_device_placement=True))
    # Runs the op.
    print(sess.run(c))
```
device placement

- types of devices
- not all operations can be done in GPU (ops vs. kernels)

```
# Creates a graph.
c = []
for d in ['/device:GPU:2', '/device:GPU:3']:
    with tf.device(d):
        a = tf.constant([1.0, 2.0, 3.0, 4.0, 5.0, 6.0], shape=[2, 3])
        b = tf.constant([1.0, 2.0, 3.0, 4.0, 5.0, 6.0], shape=[3, 2])
        c.append(tf.matmul(a, b))
    with tf.device('/cpu:0'):
        sum = tf.add_n(c)
# Creates a session with log_device_placement set to True.
sess = tf.Session(config=tf.ConfigProto(log_device_placement=True))
# Runs the op.
print(sess.run(sum))
```

recall that GPUs are co-processors, everything exists first in RAM
data parallelism

- data parallelism through batches of data
- model + batch needs to fit in GPU memory
model parallelism

- harder to program
- needs sync
- GPU mem holds partial model and data batches
model parallelism

- harder to program
- needs sync
- GPU mem holds partial model and data batches

YOUR MATH MUST SUPPORT THE PARTITIONING OF YOUR COMPUTATIONAL GRAPH!!
distributed computing

data is expensive to move!!!!
in/between graph replication, asynch/synch, etc.

```python
with tf.device("/job:ps/task:0"):
    weights_1 = tf.Variable(...) 
    biases_1 = tf.Variable(...)

with tf.device("/job:ps/task:1"):
    weights_2 = tf.Variable(...) 
    biases_2 = tf.Variable(...)

with tf.device("/job:worker/task:7"):
    input, labels = ...
    layer_1 = tf.nn.relu(tf.matmul(input, weights_1) + biases_1)
    logits = tf.nn.relu(tf.matmul(layer_1, weights_2) + biases_2)
    # ...
    train_op = ...

with tf.Session("grpc://worker7.example.com:2222") as sess:
    for _ in range(10000):
        sess.run(train_op)
```
distributed computing

data is expensive to move!!!!
in/between graph replication, asynch/synch, etc.

```python
with tf.device("/job:ps/task:0"):
    weights_1 = tf.Variable(

with tf.device("/job:worker/task:7"):
    input, labels = ...
    layer_1 = tf.nn.relu(tf.matmul(input, weights_1) + biases_1)
    logits = tf.nn.relu(tf.matmul(layer_1, weights_2) + biases_2)
    # ...
    train_op = ...

with tf.Session("grpc://worker7.example.com:2222") as sess:
    for _ in range(10000):
        sess.run(train_op)
```
CNN applications

Mask R-CNN
- Object Detection
- Segmentation
CNN applications
RNNs

Long-Short Term Memory module: LSTM

long-short term memory modules used in an RNN
RNN applications

signal patterns (finance, speech, etc.)
text generation, translation

see http://karpathy.github.io/2015/05/21/rnn-effectiveness/
other stuff

- High Level API (tflearn, Keras)
- TPUs (tensor processing units)
- Theano and Torch

https://s3.amazonaws.com/rlx/streetview_detection/madrid_centro/dboard.html

https://s3.amazonaws.com/rlx/eafit_edificios/dboard_clean.html
RISE study group on TF

- Learn and discuss on TF
- Understand its applicability
- Approach problems and practical solutions
- Compete in challenges!!!!

- Open to anybody
- Python programming encouraged

https://goo.gl/Vf6v4A
thnx