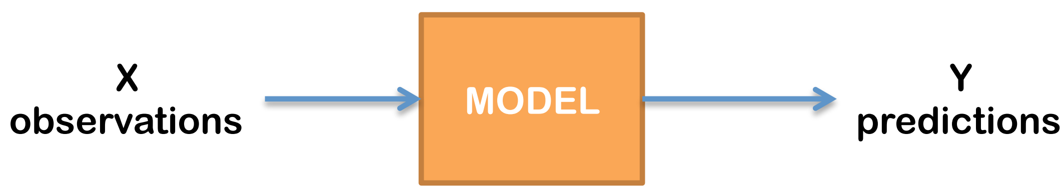
raul ramos - UdeA

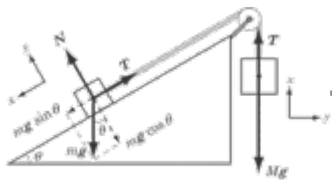
TensorFlow
symbolic computing for machine learning

TensorFlow™

this is about creating models



ANALYTIC MODELS



X
observations

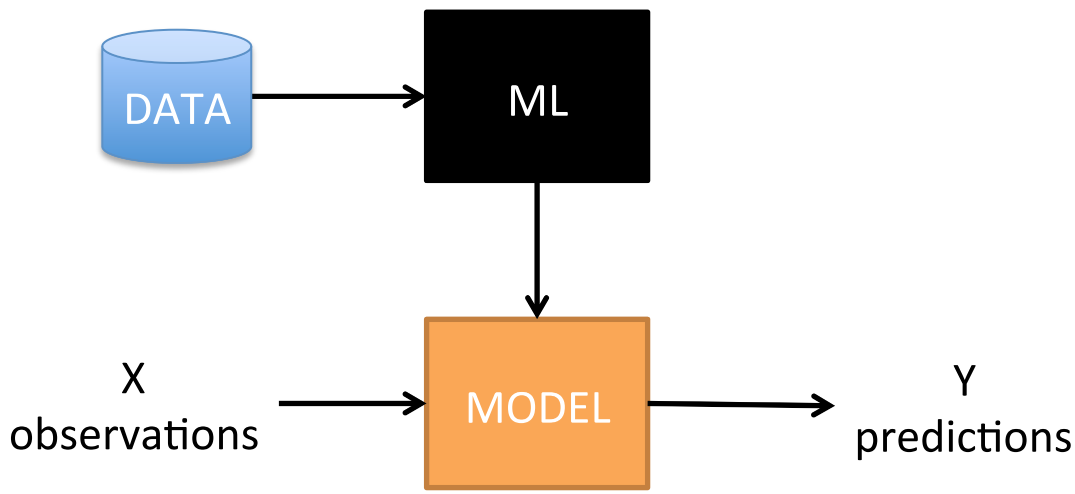
$$\frac{1}{2}Mv^2 + \frac{1}{2}mv^2 - Mgh + mgh\sin\theta = 0$$

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

$$v = \sqrt{\frac{2gh(M - m\sin\theta)}{m+M}}$$

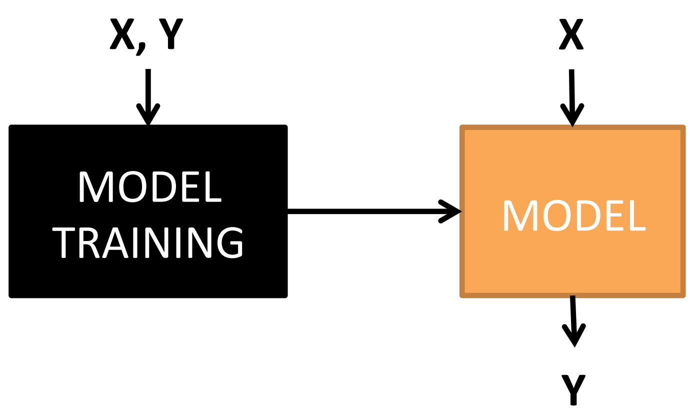
Y
predictions

MACHINE LEARNING



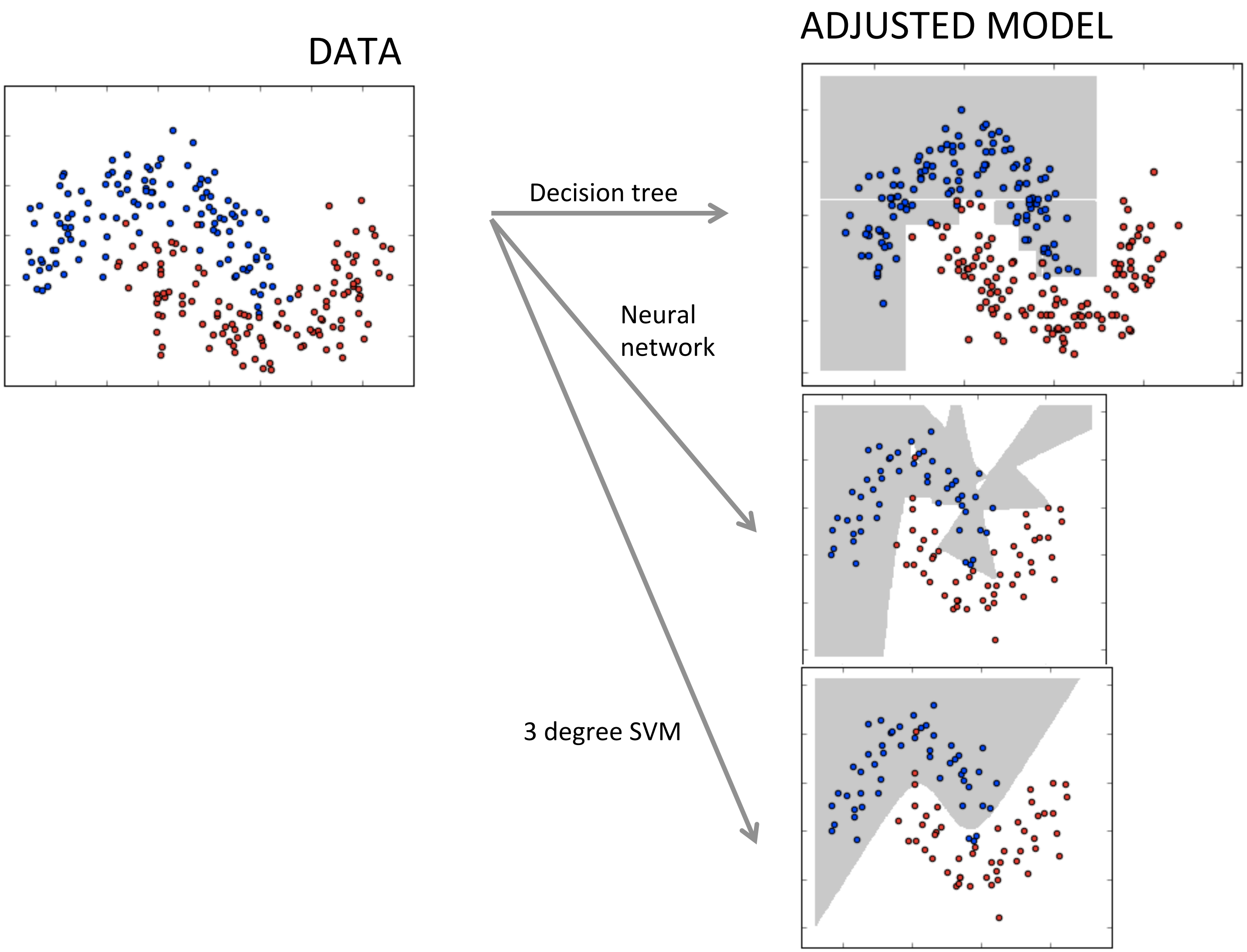
SUPERVISED LEARNING

X	Y
Patient data	Presence of pathology
Images	Objects present
Finacial data	Next price
Robot sensors	Steering action
Flight history	Landing time
Movies watched	Movies to suggest
Recorded voice	Words transcripted
Driving data	Driver profile

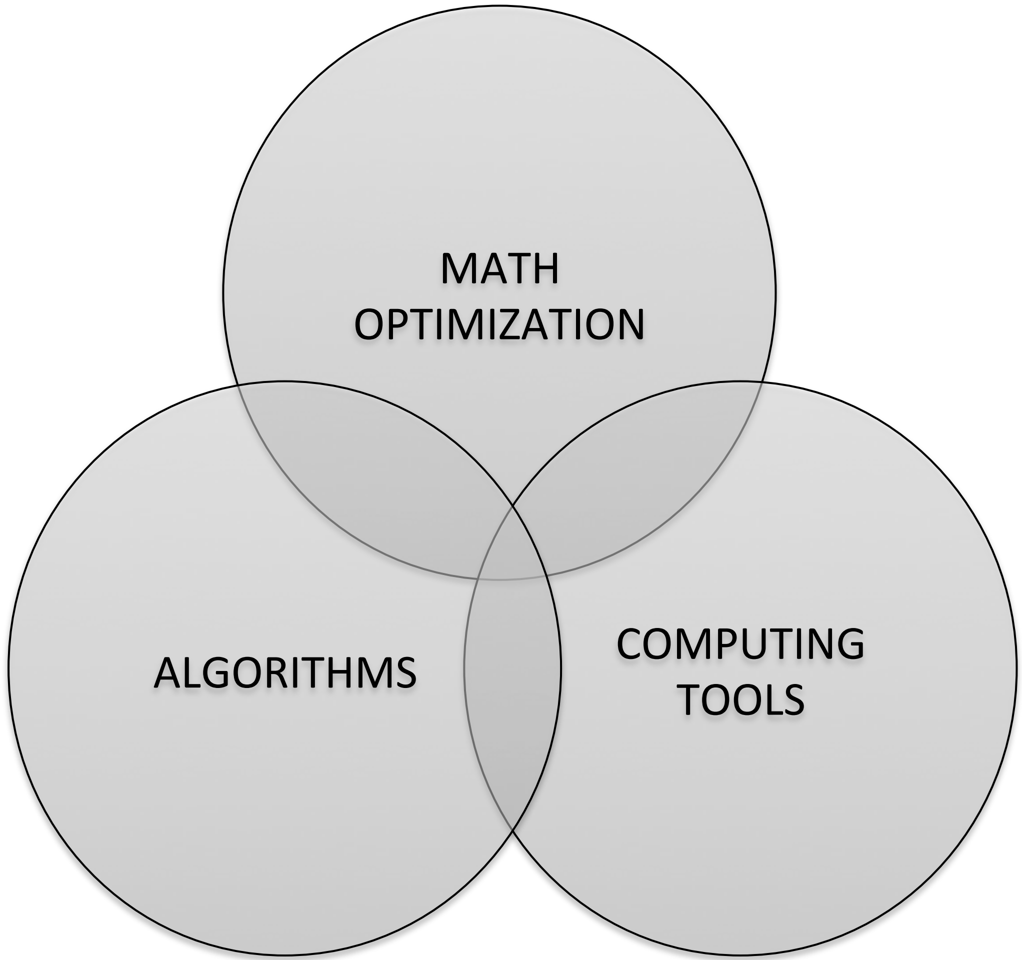


show video

FAMILIES of MODELS



MACHINE LEARNING



STANDARD FORMULATION

Tenemos un dataset de entrenamiento $\left\{(x^{(i)}, y^{(i)})\right\}$ 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 $y^{(i)}$

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

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

m: número de datos   n: número de descriptores por dato

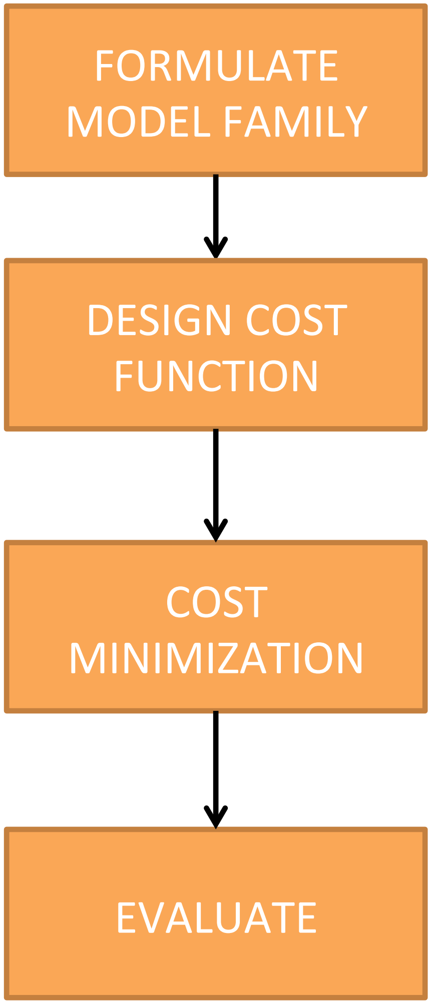
STANDARD FORMULATION

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 $y^{(i)}$

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

$$\begin{aligned} y^{(i)} \in \mathbb{R} \quad &\rightarrow \quad \textit{Regression} \\ y^{(i)} \in [0,1] \text{ } \textit{or} \text{ } [-1,1] \quad &\rightarrow \quad \textit{Classification} \end{aligned}$$

STANDARD WORKFLOW



LINEAR REGRESSION

FORMULATE MODEL FAMILY

X/Y linear relation

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

DESIGN COST FUNCTION

Avg squared error

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

$\underset{\theta}{arg\,min}\ J(\theta)$

Gradient descent
(or generic optimizer)

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

Closed form

$$\theta = \left(X^T X\right)^{-1} X^T Y$$

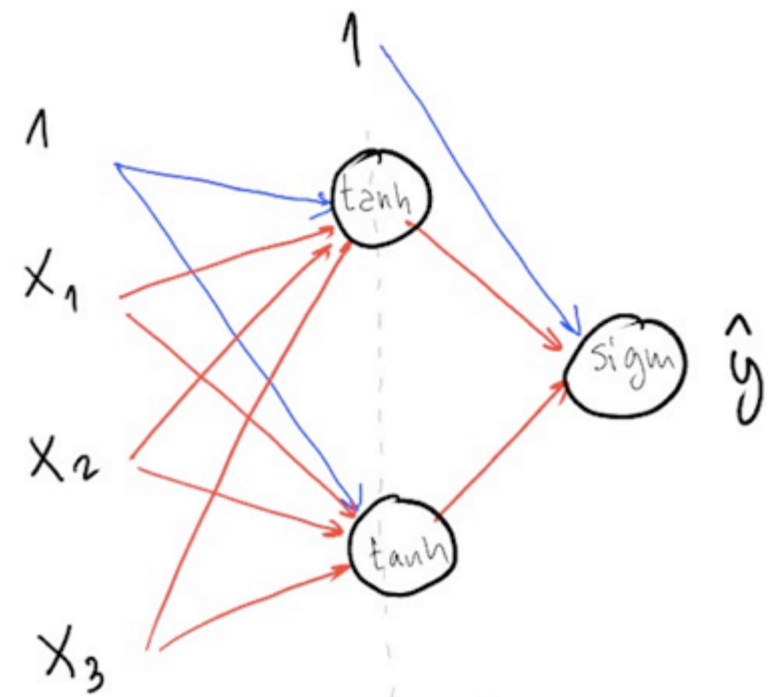
EVALUATE

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

show notebook part 1

PERCEPTRON

FORMULATE MODEL FAMILY



params: b_1 W_1 b_2 W_2

dims: 1×2 3×2 1 2×1

$$\hat{y} = \text{sigm}\left(W_2 \cdot \tanh\left(x^T W_1 + b_1\right) + b_2\right)$$

$$\mathbf{W}_1 = \begin{bmatrix} w_0^0 & w_1^0 \\ w_0^1 & w_1^1 \\ w_0^2 & w_1^2 \end{bmatrix} \qquad \mathbf{b}_1 = \begin{bmatrix} b_0 & b_1 \end{bmatrix}$$

PERCEPTRON

FORMULATE MODEL FAMILY

$$\hat{y} = g(\mathbf{W}_2 \cdot \tanh(\mathbf{x}^{\mathrm{T}} \cdot \mathbf{W}_1 + \mathbf{b}_1) + b_2)$$

↓

DESIGN COST FUNCTION

$$J(\mathbf{b}_1, b_2, \mathbf{W}_1, \mathbf{W}_2) = \frac{1}{m} \sum_{i=0}^{m-1} (\hat{y}^{(i)} - y^{(i)})^2 \,.$$

↓

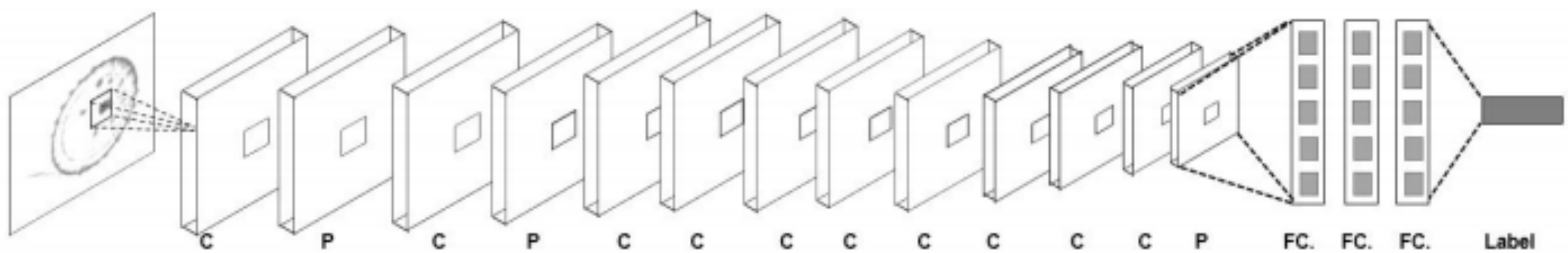
$$\arg\min_{\mathbf{b}_1, b_2, \mathbf{W}_1, \mathbf{W}_2} J(\mathbf{b}_1, b_2, \mathbf{W}_1, \mathbf{W}_2)$$

Gradient descent *(or generic optimizer)*

BACKPROPAGATION

CONVOLUTIONAL NETWORKS

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



STRUCTURE OF COST FUNCTIONS

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

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

$$\hat{y} = g(\mathbf{W}_2 \cdot \tanh(\mathbf{x}^{\mathbf{T}} \cdot \mathbf{W}_1 + \mathbf{b}_1) + b_2)$$

$$J(\mathbf{b}_1, b_2, \mathbf{W}_1, \mathbf{W}_2) = \frac{1}{m}\sum_{i=0}^{m-1}(\hat{y}^{(i)} - y^{(i)})^2 \,.$$

STRUCTURE OF COST FUNCTIONS

data

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

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

$$\hat{y} = g(\mathbf{W}_2 \cdot \tanh(\mathbf{x}^{\mathbf{T}} \cdot \mathbf{W}_1 + \mathbf{b}_1) + b_2)$$

$$J(\mathbf{b}_1, b_2, \mathbf{W}_1, \mathbf{W}_2) = \frac{1}{m} \sum_{i=0}^{m-1} (\hat{y}^{(i)} - y^{(i)})^2 \, .$$

STRUCTURE OF COST FUNCTIONS

params

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

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

$$\hat{y} = g(\mathbf{W}_2 \cdot \tanh(\mathbf{x}^{\mathbf{T}} \cdot \mathbf{W}_1 + \mathbf{b}_1) + b_2)$$

$$J(\mathbf{b}_1, b_2, \mathbf{W}_1, \mathbf{W}_2) = \frac{1}{m}\sum_{i=0}^{m-1}(\hat{y}^{(i)} - y^{(i)})^2 .$$

STRUCTURE OF COST FUNCTIONS

params

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

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

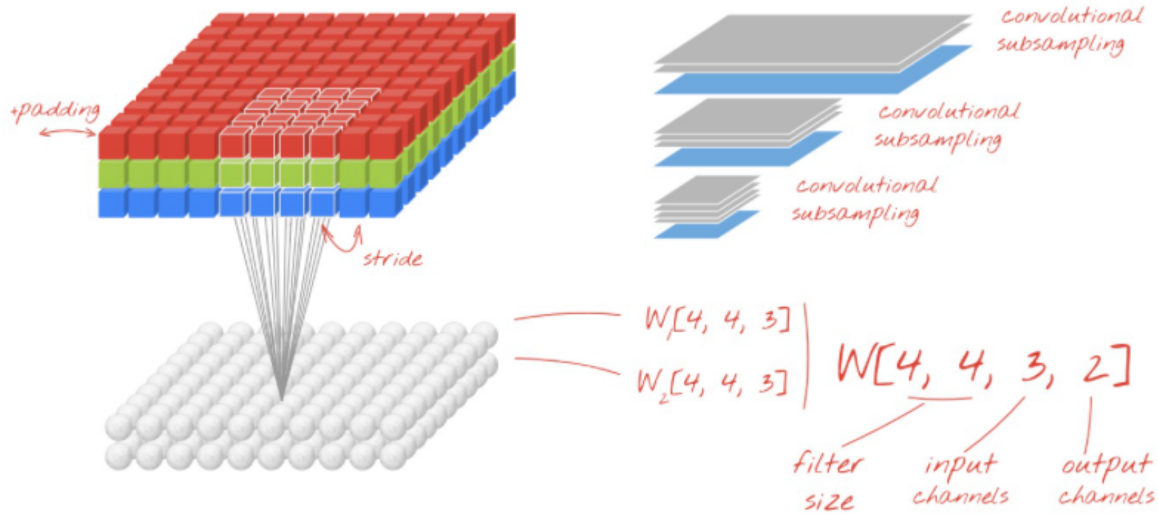
$$J(\theta) = \frac{1}{m} \sum_{i=0}^{m-1} (\hat{y}^{(i)} - y^{(i)})^2$$

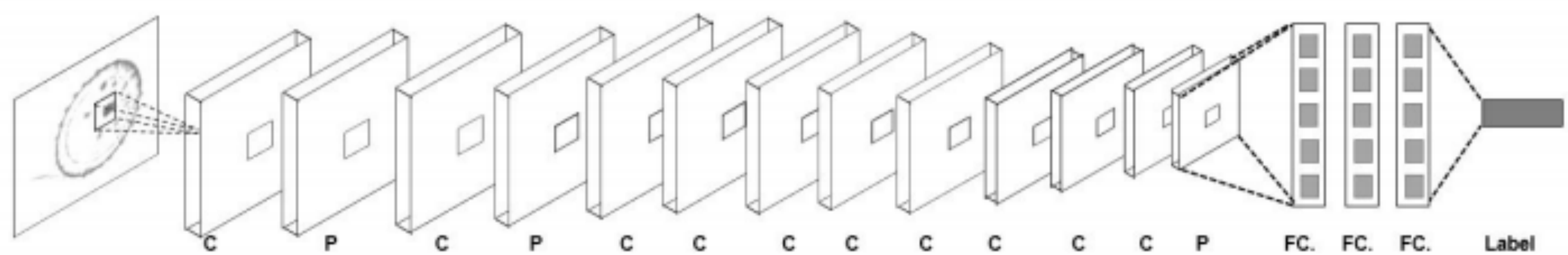
$$\hat{y} = g(\mathbf{W}_2 \cdot \tanh(\mathbf{x}^{\mathbf{T}} \cdot \mathbf{W}_1 + \mathbf{b}_1) + b_2)$$

$$J(\mathbf{b}_1, b_2, \mathbf{W}_1, \mathbf{W}_2) = \frac{1}{m} \sum_{i=0}^{m-1} (\hat{y}^{(i)} - y^{(i)})^2 \, .$$

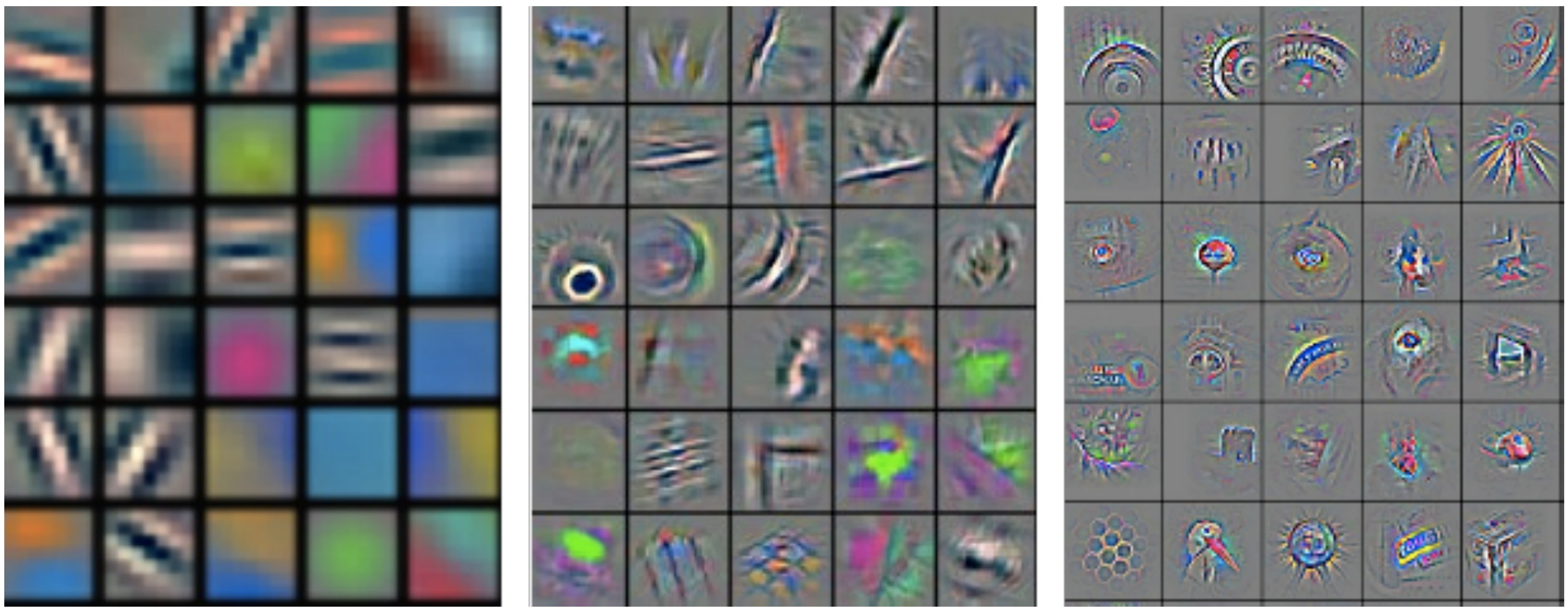
show notebook part 2

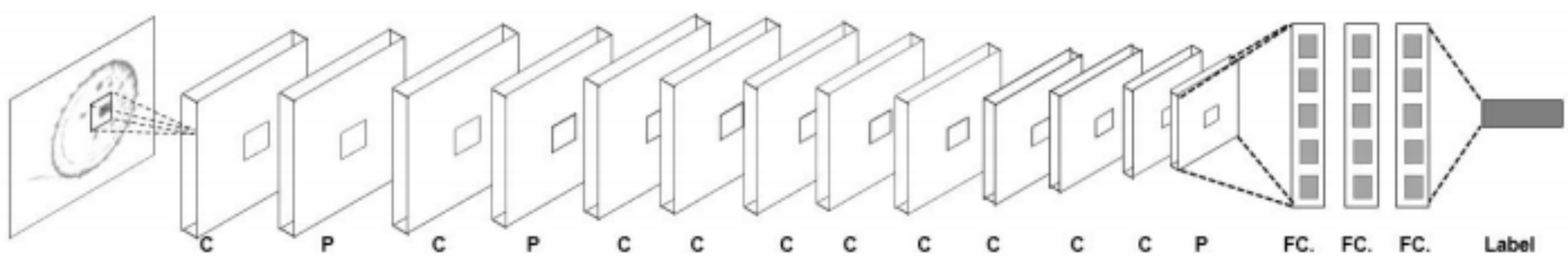
CNN


+padding
stride
convolutional
subsampling
convolutional
subsampling
convolutional
subsampling
$W_1[4, 4, 3]$
$W_2[4, 4, 3]$
W[4, 4, 3, 2]
filter size
input channels
output channels


C
P
C
P
C
C
C
C
C
C
C
C
P
FC.
FC.
FC.
Label

CNN





“small” CNN

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

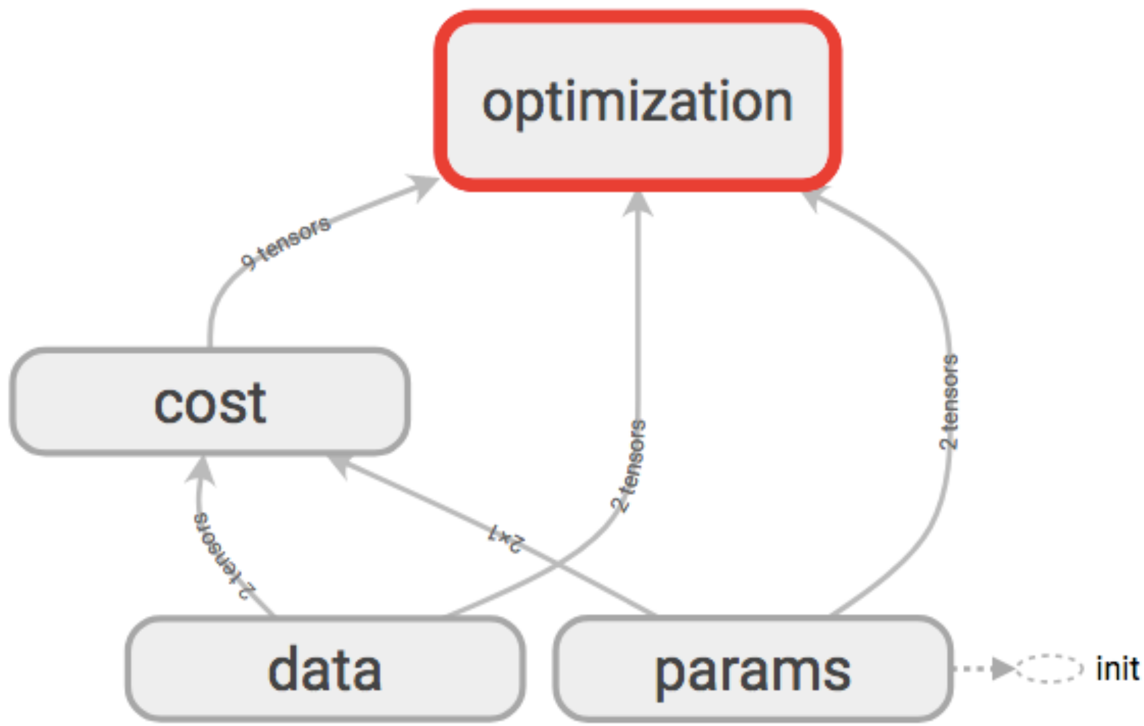
400 + 3600 + 1536 + 117600 + 2000 = 125136 tunnable params!!!

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

```
# 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

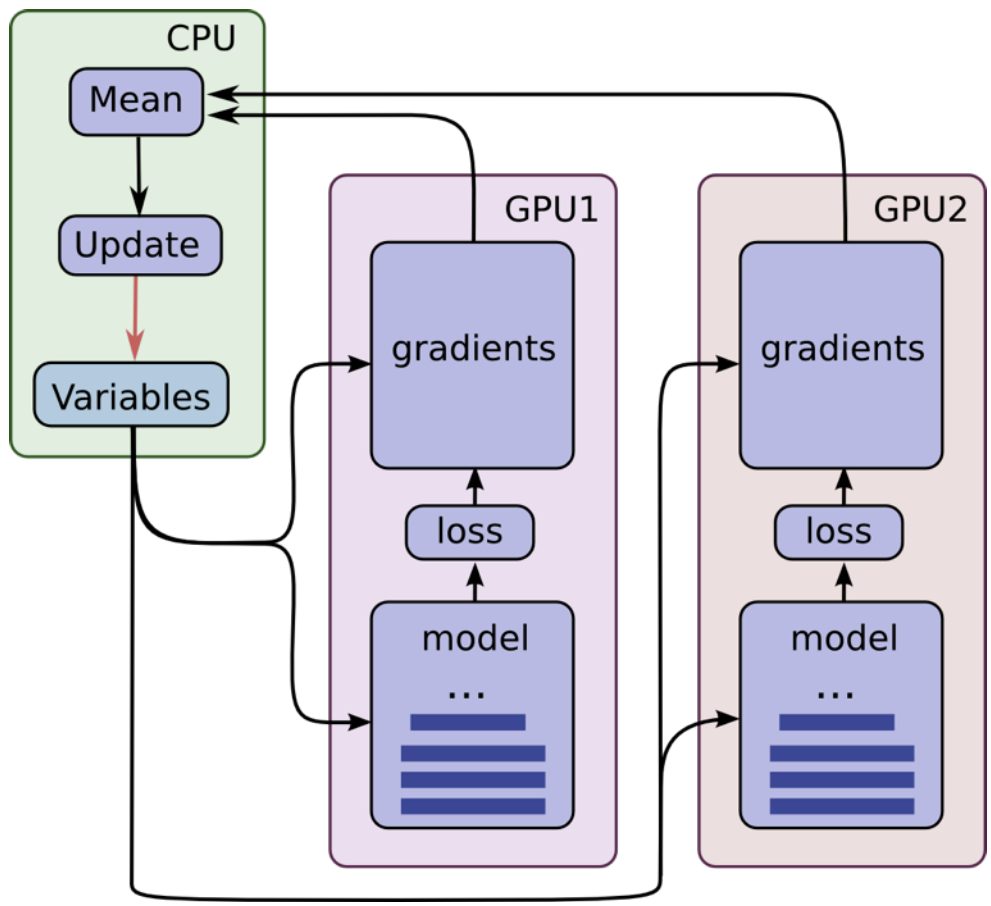
- types of devices
- not all operations can be done in GPU (ops vs. kernels)

```
# 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

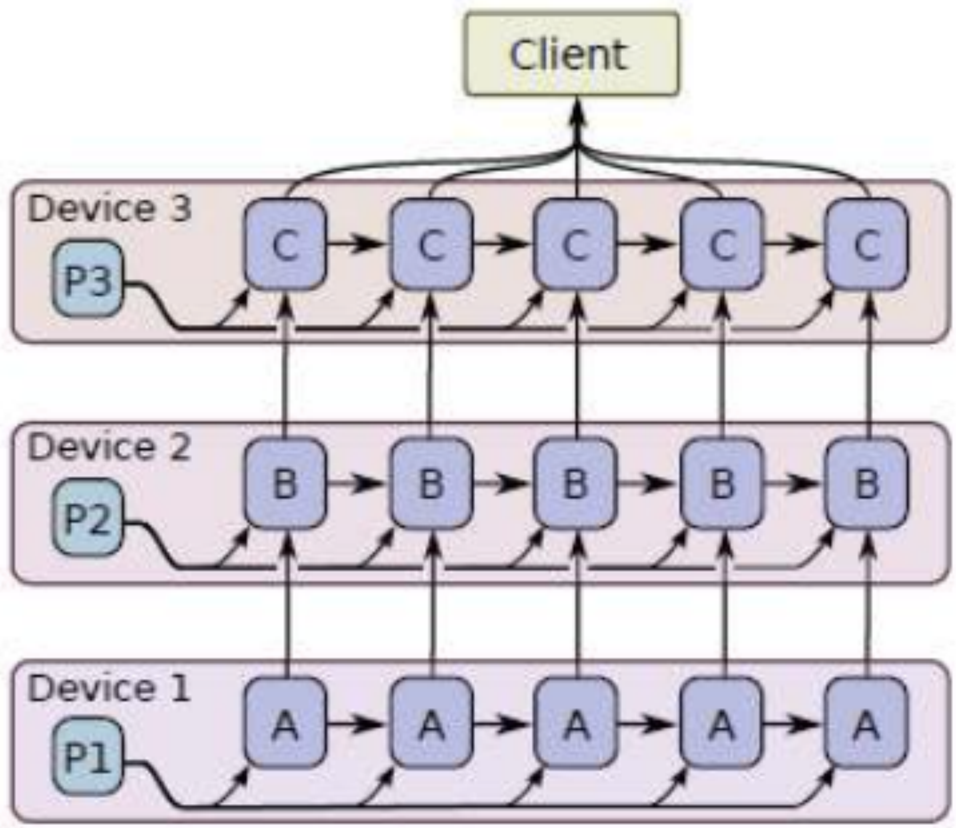
data parallelism

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



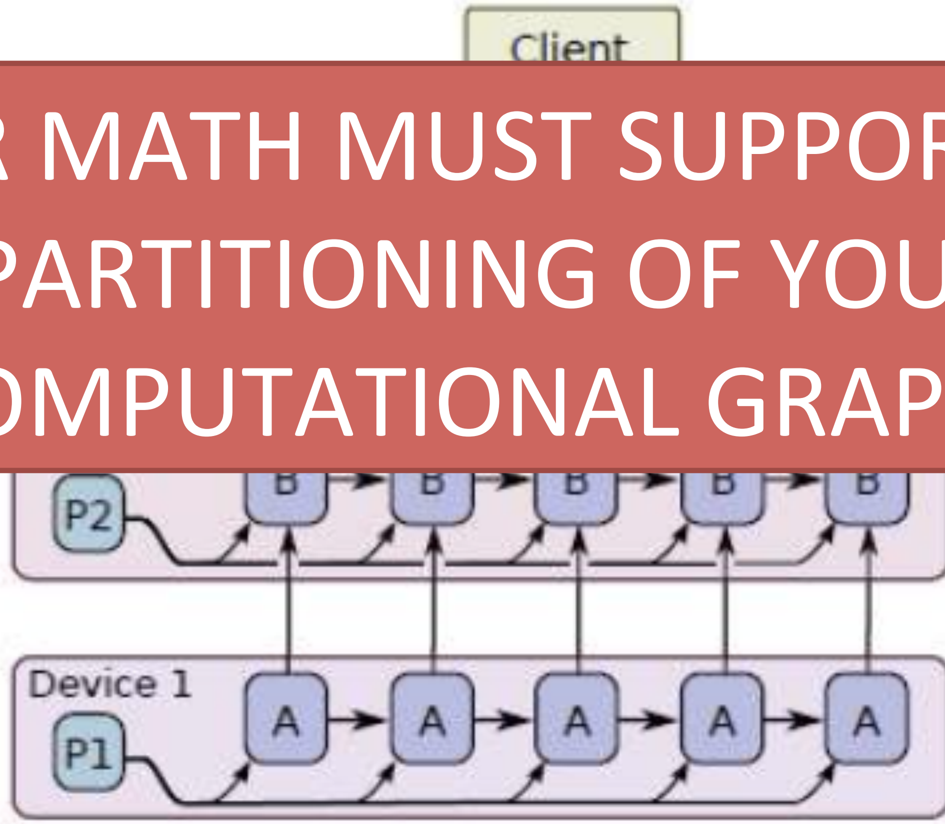
model parallelism

- 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.

```
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.

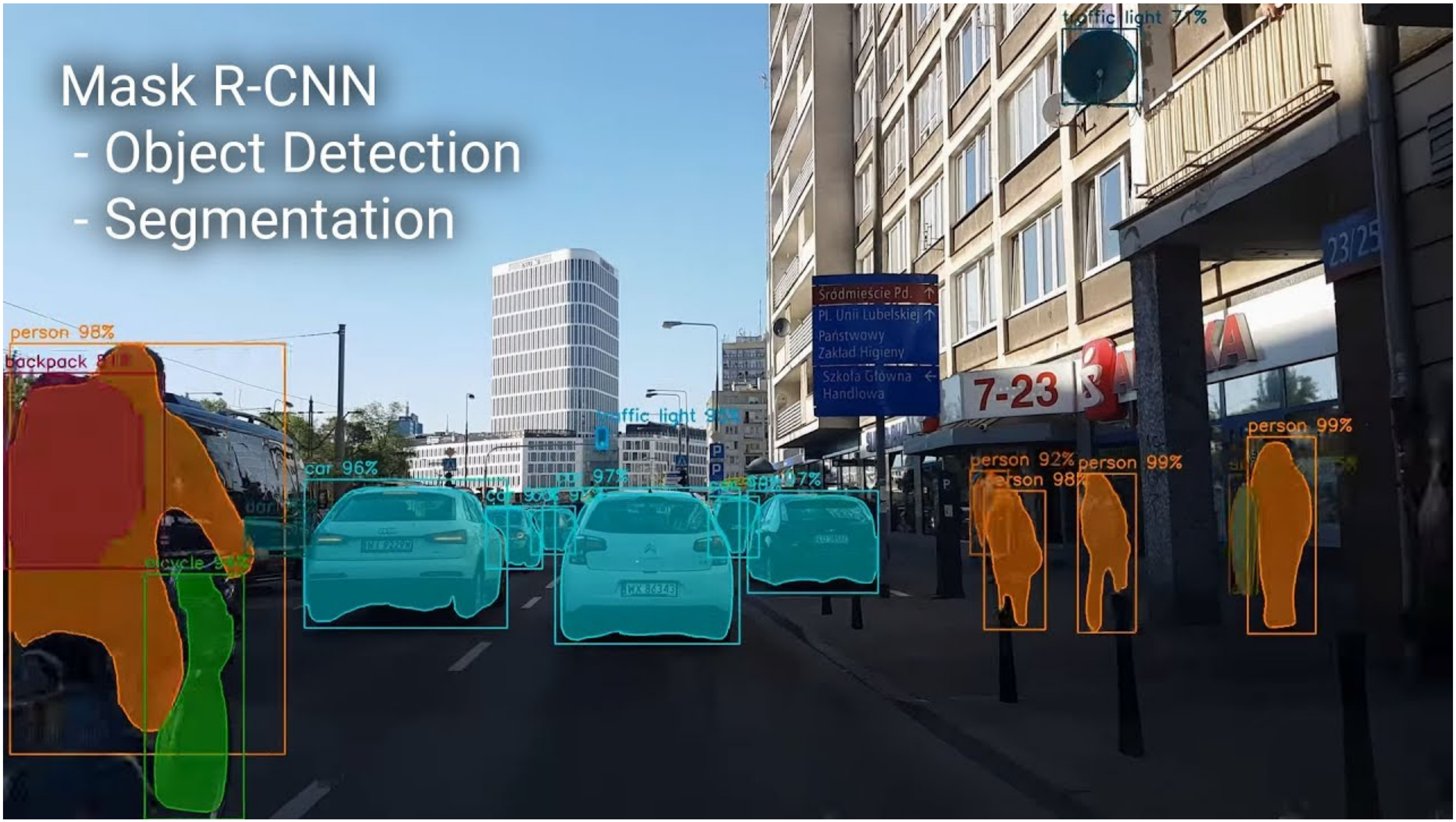
```
with tf.device("/job:ps/task:0"):
  weights_1 = tf.Variable(...)
```

THINK ON HOW DATA FLOWS WITHIN YOUR COMPUTATIONAL GRAPH

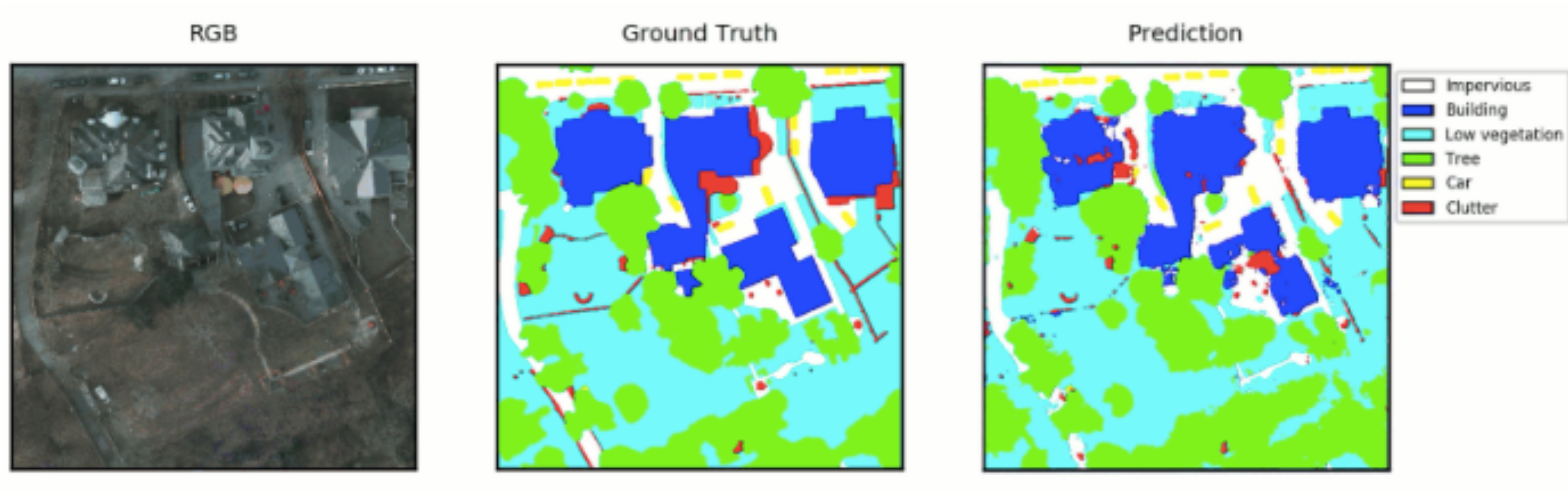
```
with tf.device("/job:worker/task:7"):
  input, labels = ...
  layer_1 = tf.nn.relu(tf.matmul(input, weights_1) + biases_1)
  logits = tf.nn.relu(tf.matmul(layer_1, weights_2) + biases_2)
  # ...
  train_op = ...

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

CNN applications

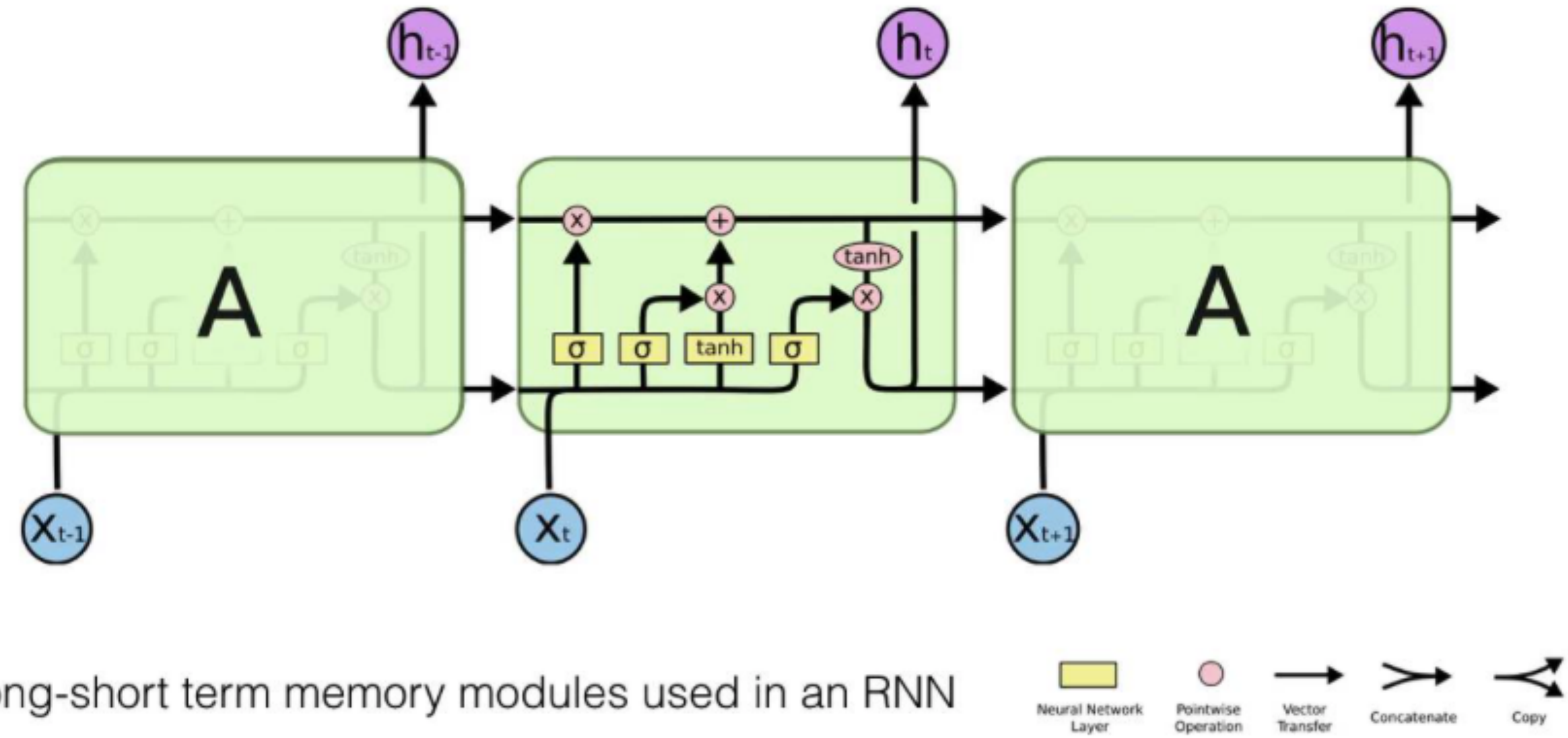


CNN applications



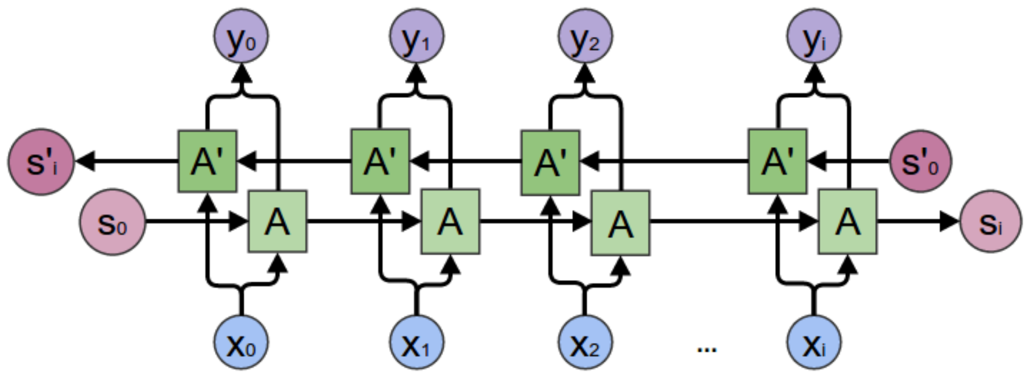
RNNs

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