Tensorflow, Part A

MSAIL 2017

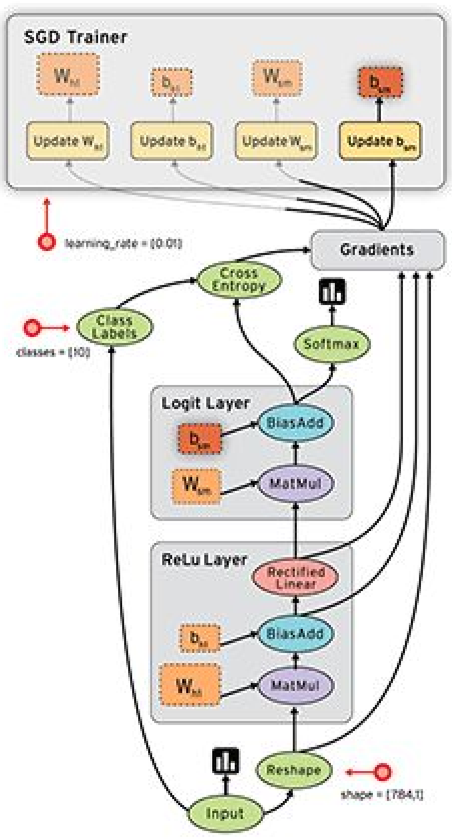
Ankit Goila, Uriah Israel, Sam Tenka

What is Tensorflow?

→ a less interpreted, more compiled version of numpy
→ Google's "library for … computation using … graphs"
→ See www.tensorflow.org/tutorials/

Why use Tensorflow?

→ fast: supports GPU computation if available
→ makes experimenting with deep neural networks easy

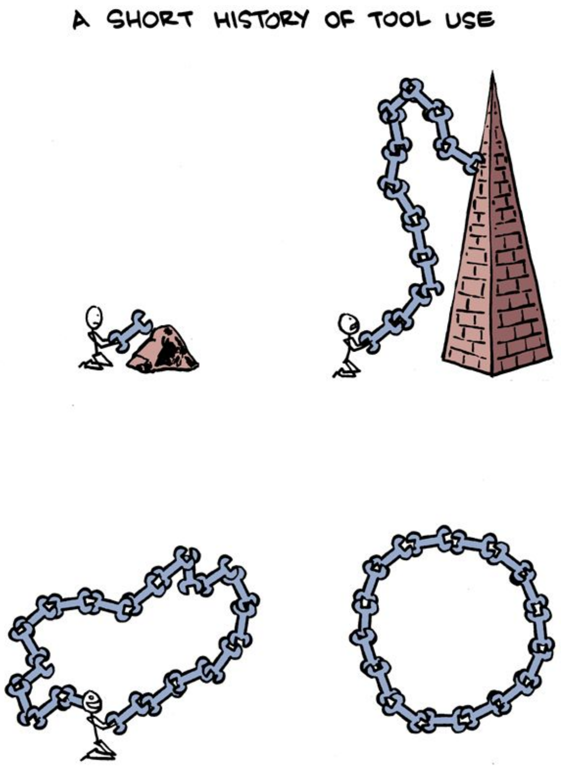


Basics: Graphs, Sessions, Ops, Variables

A brief history of automation:

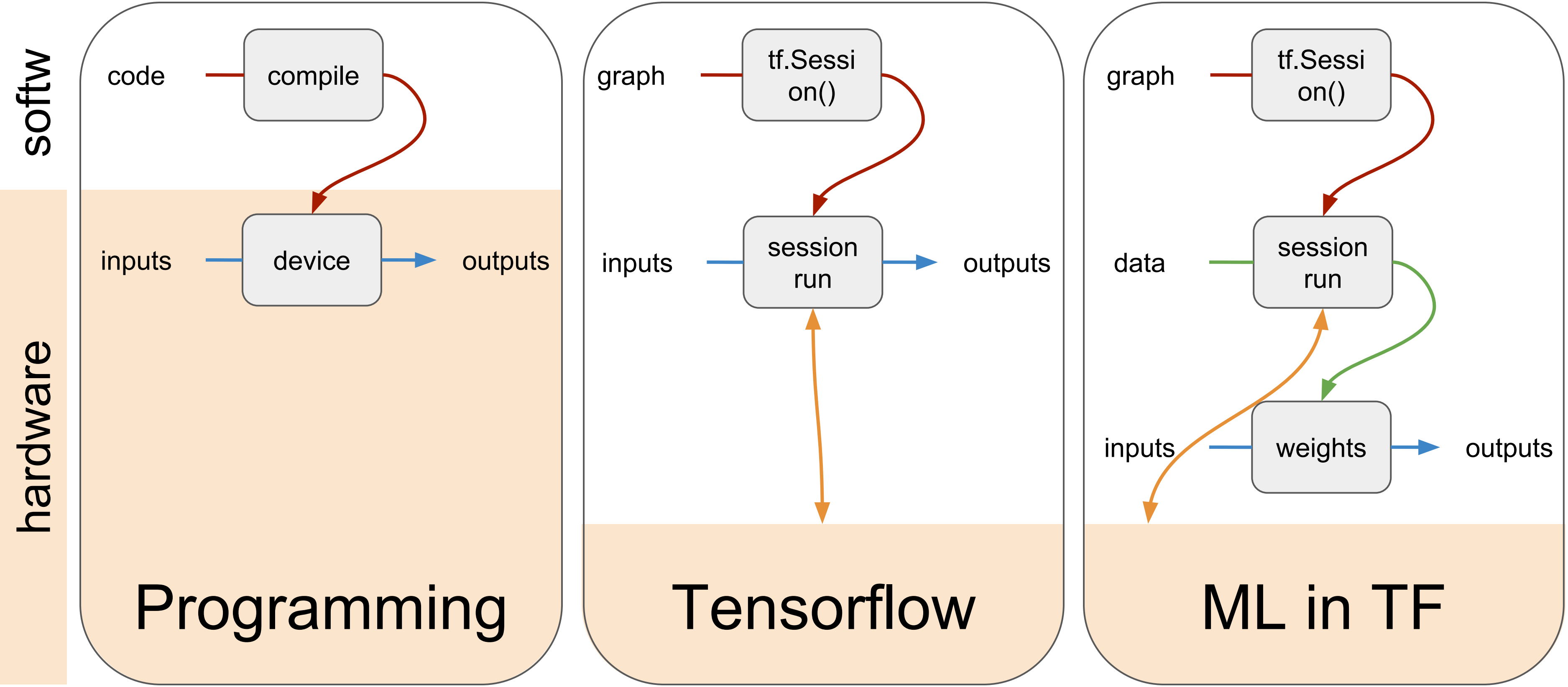
0. Manually do X
1. Manually write program to automatically do X
2. Manually write program
   to automatically write program
   to automatically do X

Machine learning is meta-programming!



Graphs are the level-2 programs. Graphs are realized in **Sessions**, which execute the level-2 programs to construct the level-1 programs.

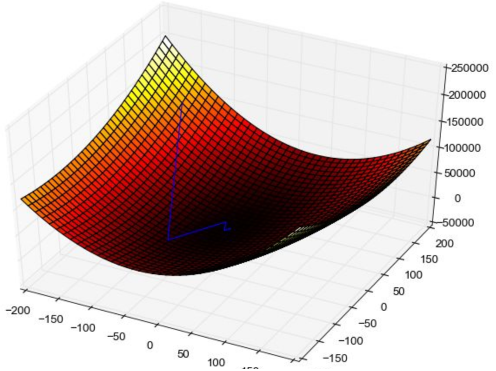
Basics: Graphs, Sessions, Ops, Variables



Basics: Manual Gradient Descent

Now we create our first Machine Learning model: linear regression!

Data: ((x[i], y[i]) for i in I) with each x[i] of a length-D vector and each y[i] a scalar.
Model: Estimate y[i] ~ **w**·x[i]. Our goal is to determine a good **w**.
Loss: Try to minimize mean square error: L = average((**w**·x[i] - y[i])**2 for i in I).



We randomly initialize **w**. Then we repeatedly adjust **w** to become more accurate on the data. This is stochastic gradient descent (SGD):

While unsatisfied:

w ← w + 0.001 * average((y[i] - **w**·x[i]) * x[i] for i in RandomSubset(I))

Basics: Automatic Gradient Descent

Two essential time-savers for deep learning experiments:

We manually differentiated, but Tensorflow can differentiate automatically.

We used vanilla SGD, but Tensorflow comes with some smarter optimizers.

```
#OLD:
GradPredictedOutputs = 2 * (PredictedOutputs - TrueOu...
GradWeights = tf.reduce_mean(tf.multiply(TrueInputs, ...
Update = tf.assign(Weights, Weights - LearningRate*GradWeights)
```

```
#NEW:
Update =
tf.train.GradientDescentOptimizer(LearningRate).minimize(Loss)
```

We now have the complete structure of a Tensorflow deep learning program. We just need to replace the linear regressor with a deep neural network! Let us do shallow networks first...

Deep Learning: A Fully Connected Classifier

Two essential time-savers for deep learning experiments:

- Automatic Differentiation
- Built-In Optimizers

Let us use the above to classify images!

Data: ((x[i], y[i]) for i in I)

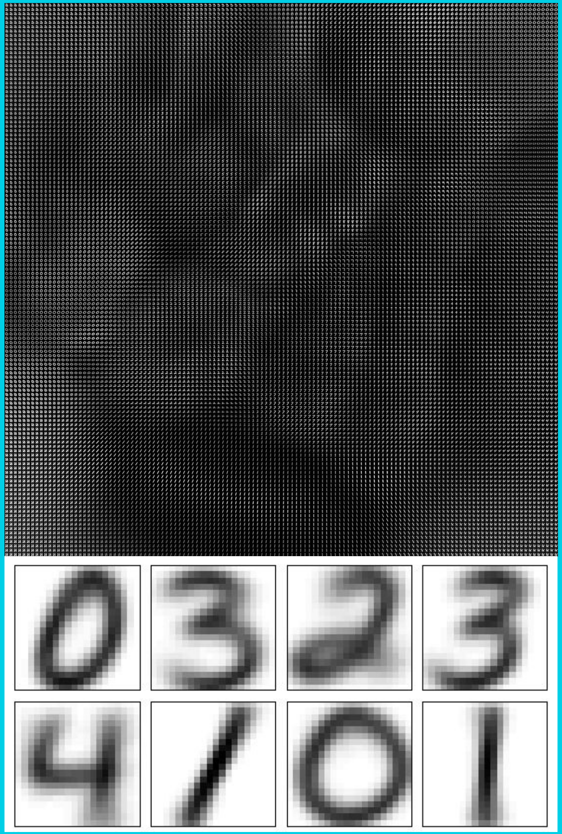
x[i] has shape (28*28) and y[i] has shape (10,)

Model: Estimate y[i] ~ Normalize(exp(**w**·x[i])).

w has shape (10, 28*28).

Loss: Minimize cross-entropy:

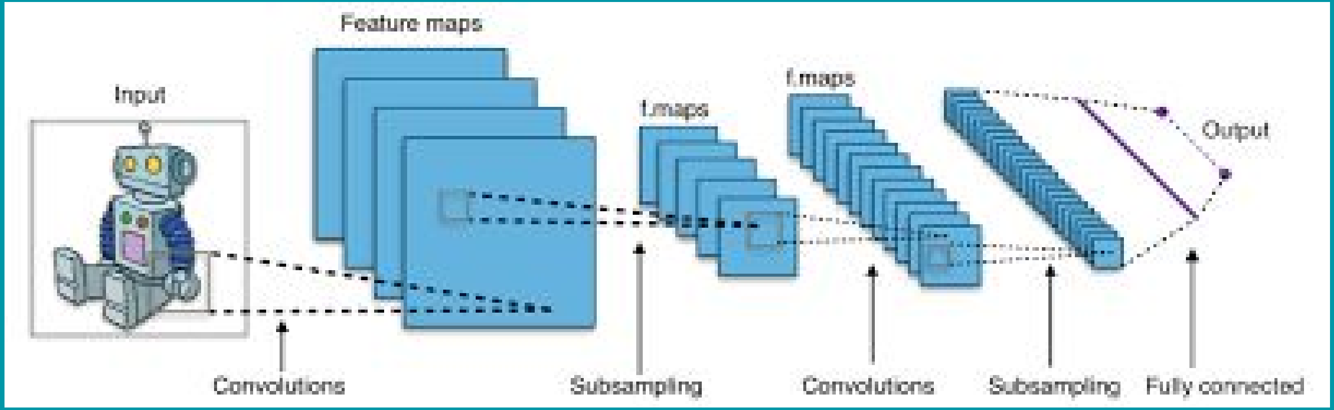
L = - average(log(guess[i])·y[i] for i in I).



Deep Learning: A Convolutional Classifier

Hmm… around 91% accuracy isn't bad: compare to the don't-look-at-the-data baseline of only 10% accuracy.

But can we do better? Let us build a convolutional classifier...



Questions?

- Check out the exercises in README.markdown. Is it clear how to approach them?