How Jupyter Notebook Helped Us Teach Deep Learning to 100,000 Students

Rachel Thomas
fast.ai & USF Data Institute
@math_rachel
My Background

Swarthmore: Math + CS
Duke: Math PhD

Quant

Uber Data Scientist & Engineer

fast.ai Researcher & Founder

Hackbright Instructor

@math_rachel
data science blog: fast.ai
https://medium.com/@rachelho
Deep Learning == multi-layered neural networks
Baidu’s Deep-Learning System is better at English and Mandarin Speech Recognition than most people.

China’s leading Internet-search company, Baidu, has developed a voice system that can recognize English and Mandarin speech better than people, in some cases.
Growing Use of Deep Learning at Google

# of directories containing model description files

Across many products/areas:
- Android
- Apps
- Drug discovery
- Gmail
- Image understanding
- Maps
- Natural language understanding
- Photos
- Robotics research
- Speech
- Translation
- YouTube
- ... many others ...

Unique project directories

2400
2012-Q1, Q2, Q3, Q4
2013-Q1, Q2, Q3, Q4
2014-Q1, Q2, Q3, Q4
2015-Q1, Q2, Q3, Q4
2016-Q1, Q2

Time

@math_rachel
To gain some intuition for how the BPTT algorithm behaves, we provide an example of how to compute gradients by BPTT for the RNN equations above (equation 10.8 and equation 10.12). The nodes of our computational graph include

$$\frac{\partial L}{\partial L(t)} = 1. \quad (10.17)$$

In this derivation we assume that the outputs $o(t)$ are used as the argument to the softmax function to obtain the vector $\hat{y}$ of probabilities over the output. We also assume that the loss is the negative log-likelihood of the true target $y(t)$ given the input so far. The gradient $\nabla_{o(t)}L$ on the outputs at time step $t$, for all $i, t$, is as follows:

$$\left(\nabla_{o(t)}L\right)_i = \frac{\partial L}{\partial o_i(t)} = \frac{\partial L}{\partial L(t)} \frac{\partial L(t)}{\partial o_i(t)} = \hat{y}_i(t) - 1_{i, y(t)}. \quad (10.18)$$

We work our way backwards, starting from the end of the sequence. At the final time step $\tau$, $h(\tau)$ only has $o(\tau)$ as a descendent, so its gradient is simple:

$$\nabla h(\tau) L = V^\top \nabla_{o(\tau)}L. \quad (10.19)$$

We can then iterate backwards in time to back-propagate gradients through time, from $t = \tau - 1$ down to $t = 1$, noting that $h(t)$ (for $t < \tau$) has as descendents both $o(t)$ and $h(t+1)$. Its gradient is thus given by

$$\nabla h(t) L = \left(\nabla h(t+1) L\right)^\top + \left(\nabla o(t) L\right)^\top \left(\nabla o(t) L\right) \quad (10.20)$$

$$= W^\top \left(\nabla h(t+1) L\right) \text{diag}\left(1 - (h(t+1))^2\right) + V^\top \left(\nabla o(t) L\right) \quad (10.21)$$

Using this notation, the gradient on the remaining parameters is given by:

$$\nabla cL = \sum_t \left(\frac{\partial o(t)}{\partial c}\right)^\top \nabla_{o(t)}L = \sum_t \nabla_{o(t)}L \quad (10.22)$$

$$\nabla bL = \sum_t \left(\frac{\partial h(t)}{\partial b(t)}\right)^\top \nabla_{h(t)}L = \sum_t \text{diag}\left(1 - (h(t))^2\right) \nabla_{h(t)}L \quad (10.23)$$

$$\nabla vL = \sum_t \sum_i \left(\frac{\partial o(t)}{\partial o_i(t)}\right)^\top \nabla_{o_i(t)}L = \sum_t \left(\nabla_{o(t)}L\right) h(t)^\top \quad (10.24)$$

$$\nabla wL = \sum_t \sum_i \left(\frac{\partial L}{\partial h_i(t)}\right) \nabla_{h_i(t)}L = \sum_t \text{diag}\left(1 - (h(t))^2\right) \left(\nabla_{h(t)}L\right) h(t-1)^\top \quad (10.25)$$

$$\nabla uL = \sum_t \sum_i \left(\frac{\partial L}{\partial h_i(t)}\right) \nabla_{h_i(t)}L = \sum_t \text{diag}\left(1 - (h(t))^2\right) \left(\nabla_{h(t)}L\right) x(t)^\top \quad (10.26)$$

@math_rachel
Welcome to fast.ai’s 7 week course, Practical Deep Learning For Coders, Part 1, taught by Jeremy Howard (Kaggle’s #1 competitor 2 years running, and founder of Enlitic). Learn how to build state of the art models without needing graduate-level math—but also without dumbing anything down. Oh and one other thing… it’s totally free!

Reducing overfitting

Resnet

```
In [4]: import resnet50; reload(resnet50)
from resnet50 import Resnet50

In [5]: rn0 = Resnet50(include_top=False).model

In [7]: rn0.output_shape
```

IF YOU CAN CODE, YOU CAN DO DEEP LEARNING
Sorting 2 Tons of Legos, Jacques Mattheij

@math_rachel
Splunk and Tensorflow for Security: Catching the Fraudster with Behavior Biometrics
Cats vs Dogs

@math_rachel
Use a pretrained model with our **Vgg16** class

Setup

The punchline: state of the art custom model in 6 lines of code

Here's everything you need to do to get >97% accuracy on the Dogs vs Cats dataset - we won't analyze how it works behind the scenes yet, since at this stage we're just going to focus on the minimum necessary to actually do useful work.

```python
In [ii]:

  vgg = Vgg16()

  # Grab a few images at a time for training and validation.
  batches = vgg.get_batches(path+'train', batch_size=batch_size)
  val_batches = vgg.get_batches(path+'valid', batch_size=batch_size*2)
  vgg.finetune(batches)
  vgg.fit(batches, val_batches, nb_epoch=1)
```

Found 160 images belonging to 2 classes.
Found 40 images belonging to 2 classes.
Epoch 1/1
160/160 [==============================] - 121s - loss: 1.6930 - acc: 0.5375 - val_loss: 0.4402 - val_acc: 0.8000
“I personally fell into the habit of watching the lectures too much and googling definitions / concepts / etc too much, without running the code. At first I thought that I should read the code quickly and then spend time researching the theory behind it...

“In retrospect, I should have spent the majority of my time on the actual code in the notebooks instead, in terms of running it and seeing that goes into it and what comes out of it”
As you can see, the labels for each image are an array, containing a 1 in the first position if it's a cat, and in the second position if it's a dog. This approach to encoding categorical variables, where an array containing just a single 1 in the position corresponding to the category, is very common in deep learning. It is called one hot encoding.

We can now pass the images to Vgg16's predict() function to get back probabilities, category indexes, and category names for each image's VGG prediction.
In [900]:
plt.imshow(to_plot(val[0]))

In [24]:
cm = get_cm2(inp, 4)

In [903]:
plt.figure(figsize=(10,10))
plot(val[0])
plt.imshow(cm, cmap="cool", alpha=0.5)
Stochastic Gradient Descent
My favorite extension: Collapsible Headings

Linear models with CNN features

Introduction
Linear models in keras
Train linear model on predictions

Modifying the model
Retrain last layer's linear model
Retraining more layers
An introduction to back-propagation
Training multiple layers in Keras
Thank you!

fast.ai free courses:

• Practical Deep Learning for Coders: course.fast.ai
• Cutting Edge Deep Learning for Coders: course.fast.ai/part2
• Computational Linear Algebra: github.com/fastai/numerical-linear-algebra

@math_rachel