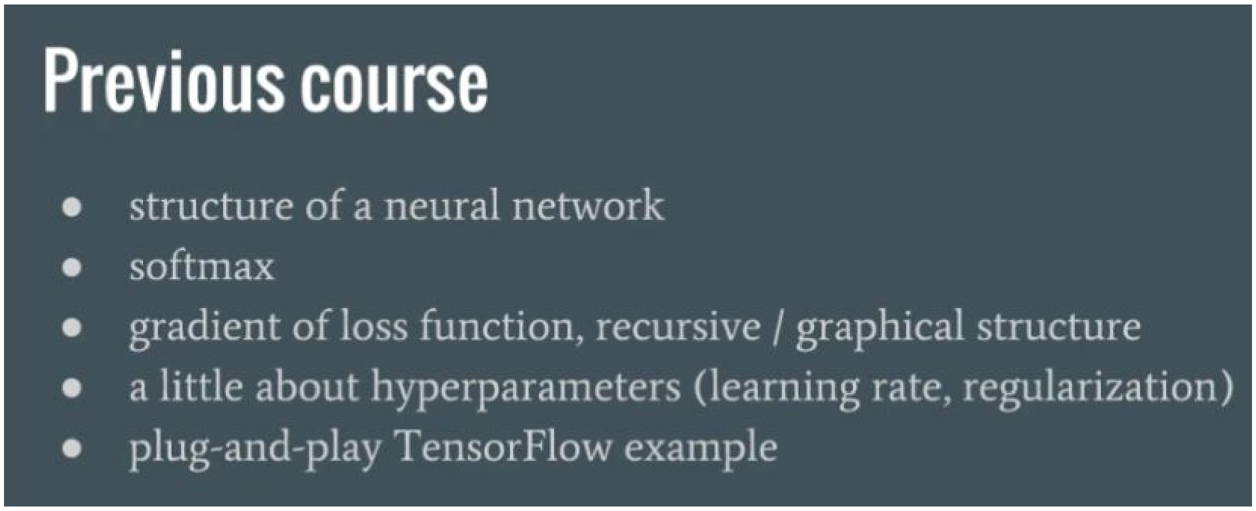
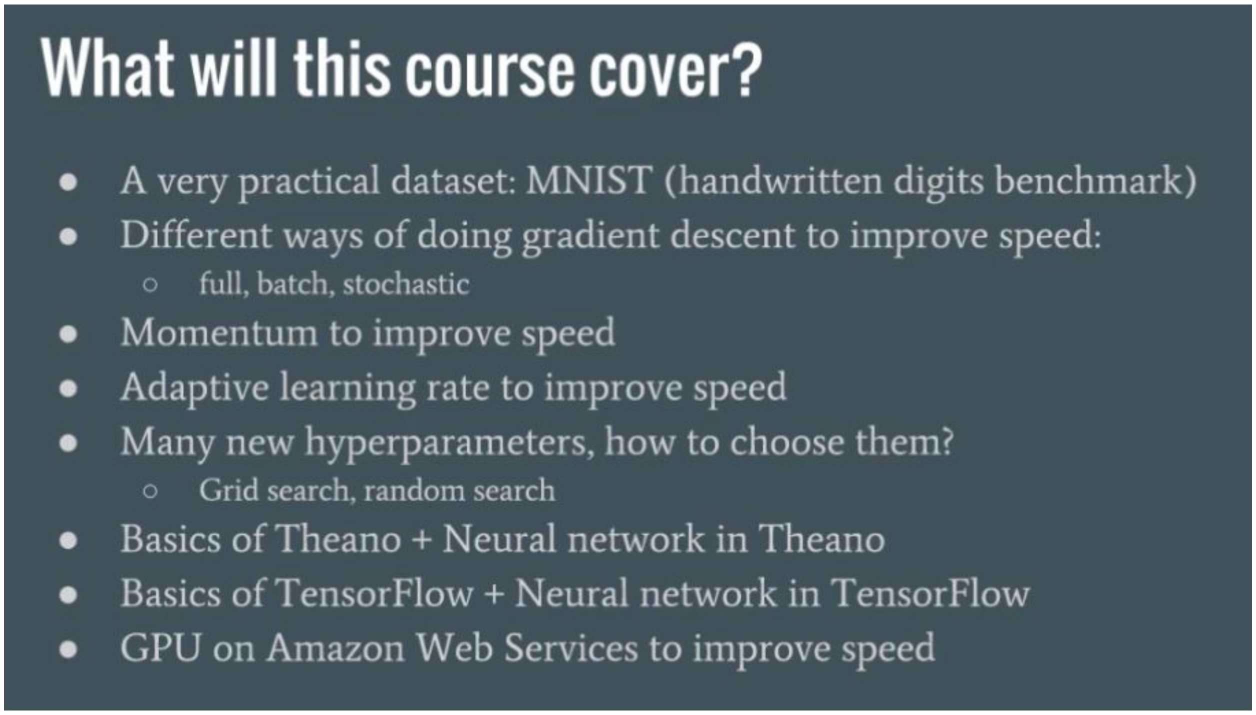
PRACTICAL DEEP LEARNING IN THEANO AND TENSORFLOW



Previous course

- structure of a neural network
- softmax
- gradient of loss function, recursive / graphical structure
- a little about hyperparameters (learning rate, regularization)
- plug-and-play TensorFlow example



What will this course cover?

- A very practical dataset: MNIST (handwritten digits benchmark)
- Different ways of doing gradient descent to improve speed:
  - full, batch, stochastic
- Momentum to improve speed
- Adaptive learning rate to improve speed
- Many new hyperparameters, how to choose them?
  - Grid search, random search
- Basics of Theano + Neural network in Theano
- Basics of TensorFlow + Neural network in TensorFlow
- GPU on Amazon Web Services to improve speed

Knowledge • 859 teams

Digit Recognizer

Wed 25 Jul 2012 Sat 31 Dec 2016 (10 months to go)

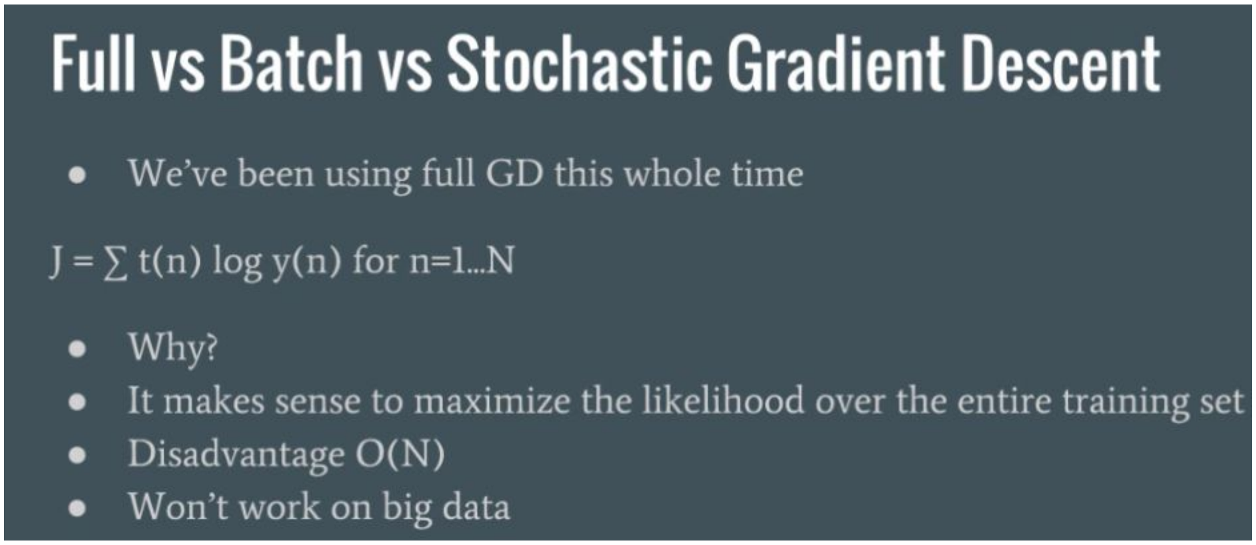
Competition Details » Get the Data » Make a submission

Classify handwritten digits using the famous MNIST data

Get started on this competition through Kaggle Scripts

The goal in this competition is to take an image of a handwritten single digit, and determine what that digit is. As the competition progresses, we will release tutorials which explain different machine learning algorithms and help you to get started.

The data for this competition were taken from the MNIST dataset. The MNIST ("Modified National Institute of Standards and Technology") dataset is a classic within the Machine Learning community that has been extensively studied. More detail about the dataset, including Machine Learning algorithms that have been tried on it and their

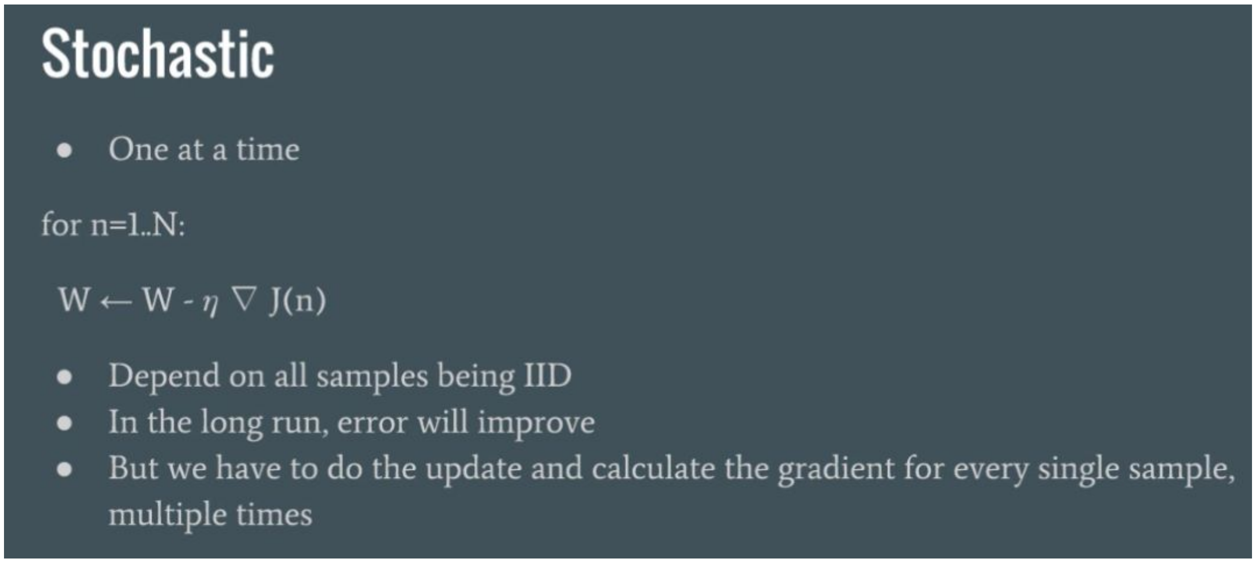


Full vs Batch vs Stochastic Gradient Descent

- We've been using full GD this whole time

$J = \sum t(n) \log y(n)$ for n=1...N

- Why?
- It makes sense to maximize the likelihood over the entire training set
- Disadvantage O(N)
- Won't work on big data

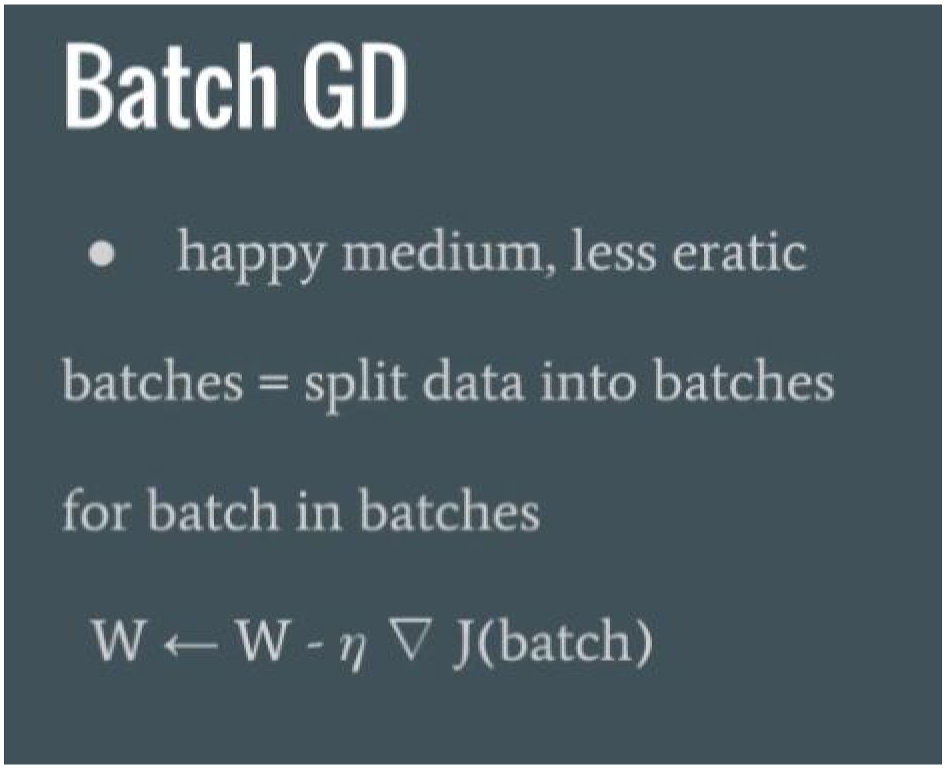


Stochastic

- One at a time

```
for n=1..N:
  $W \leftarrow W - \eta \nabla J(n)$
```

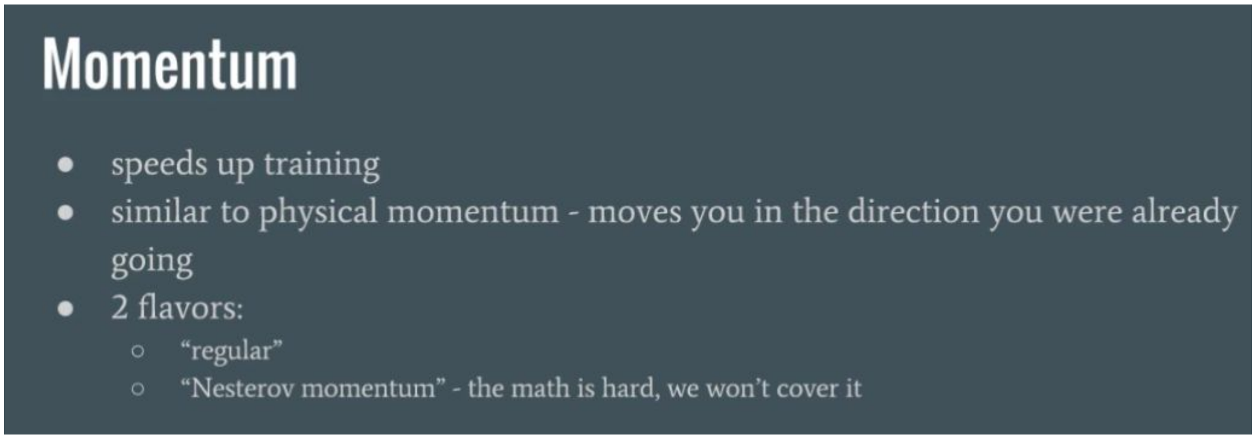
- Depend on all samples being IID
- In the long run, error will improve
- But we have to do the update and calculate the gradient for every single sample, multiple times



Batch GD

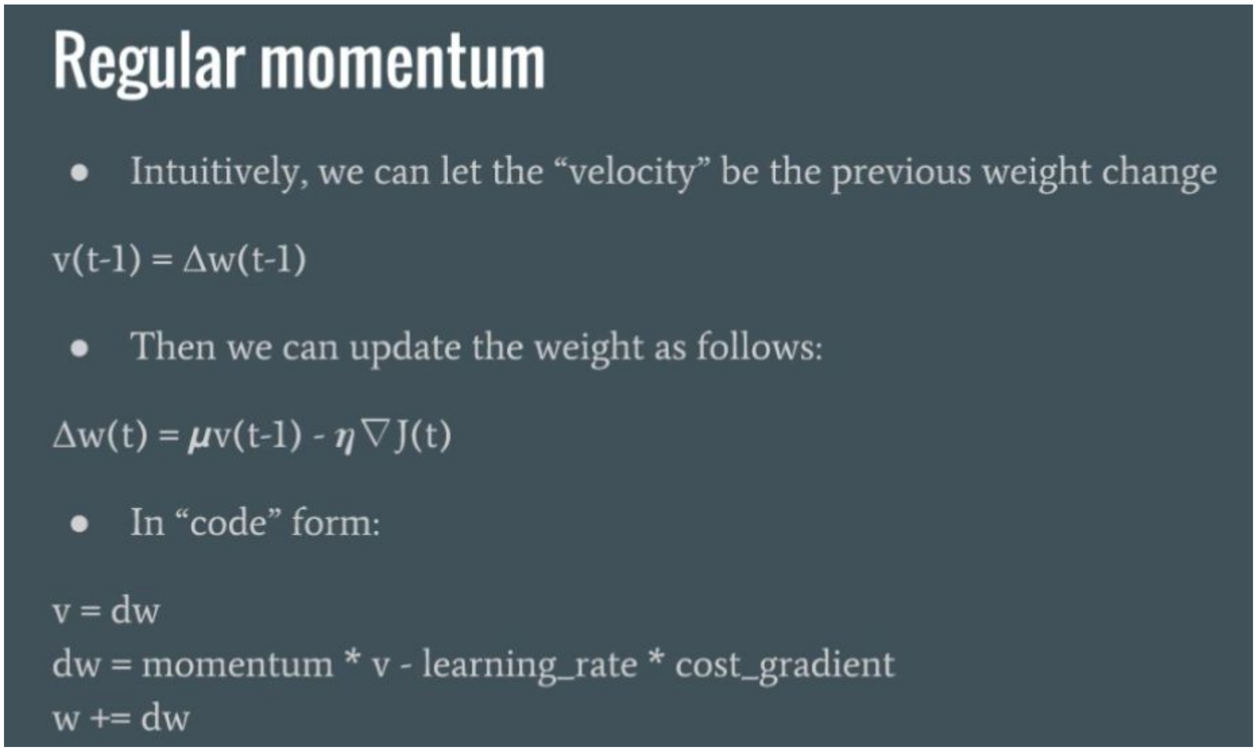
- happy medium, less eratic

```
batches = split data into batches
for batch in batches
  $W \leftarrow W - \eta \nabla J(batch)$
```



Momentum

- speeds up training
- similar to physical momentum - moves you in the direction you were already going
- 2 flavors:
  - “regular”
  - “Nesterov momentum” - the math is hard, we won’t cover it



Regular momentum

- Intuitively, we can let the “velocity” be the previous weight change

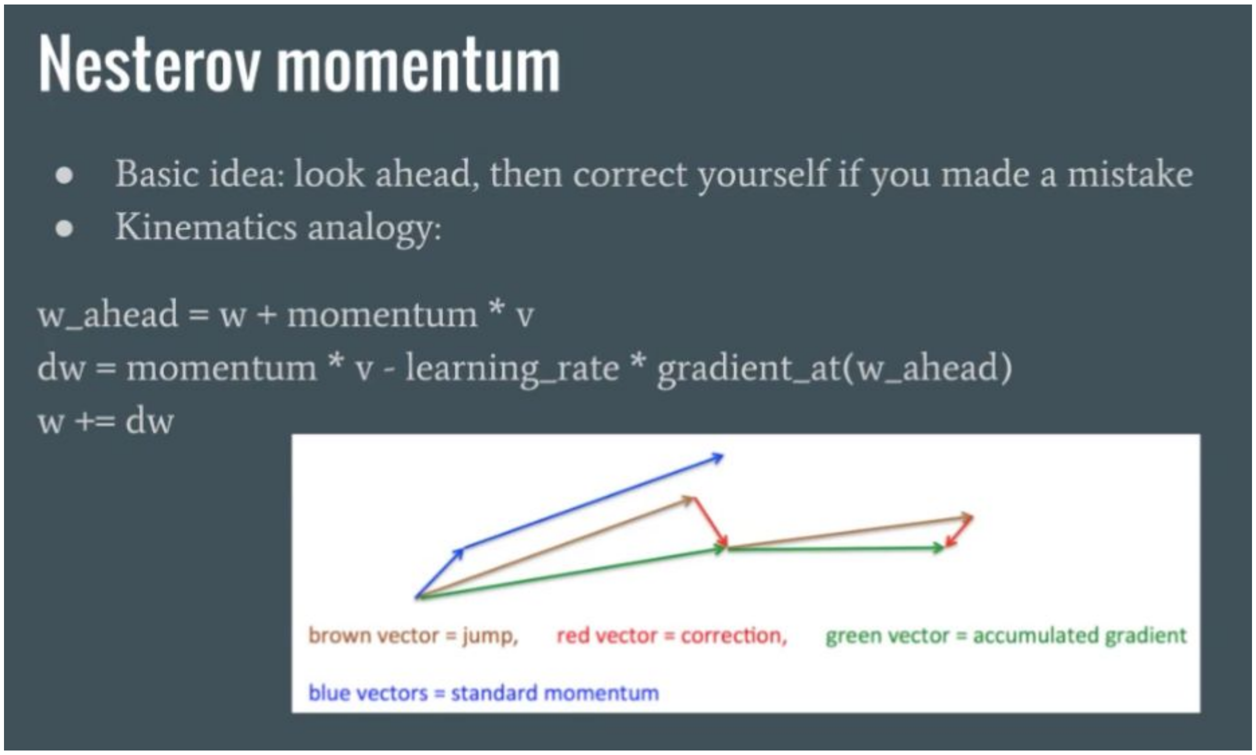
$v(t-1) = \Delta w(t-1)$

- Then we can update the weight as follows:

$\Delta w(t) = \boldsymbol{\mu} v(t-1) - \boldsymbol{\eta} \nabla J(t)$

- In “code” form:

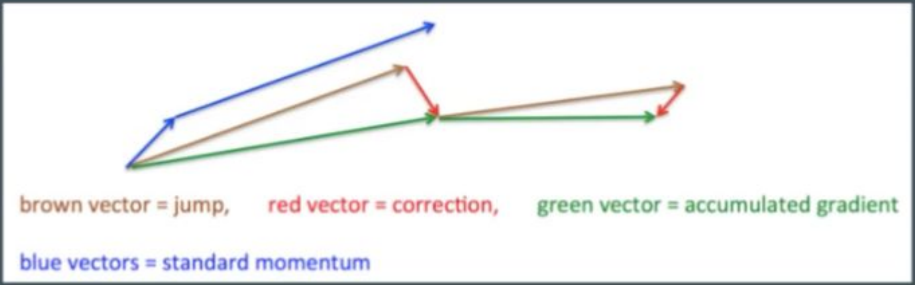
```
v = dw
dw = momentum * v - learning_rate * cost_gradient
w += dw
```

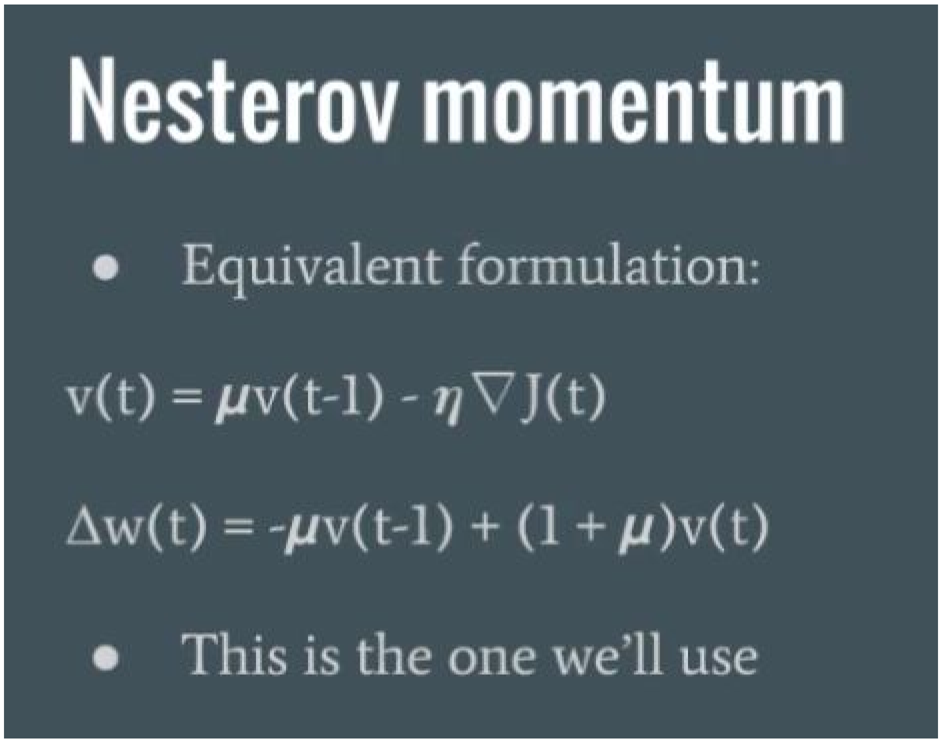


Nesterov momentum

- Basic idea: look ahead, then correct yourself if you made a mistake
- Kinematics analogy:

w_ahead = w + momentum * v
dw = momentum * v - learning_rate * gradient_at(w_ahead)
w += dw


brown vector = jump,
red vector = correction,
green vector = accumulated gradient
blue vectors = standard momentum



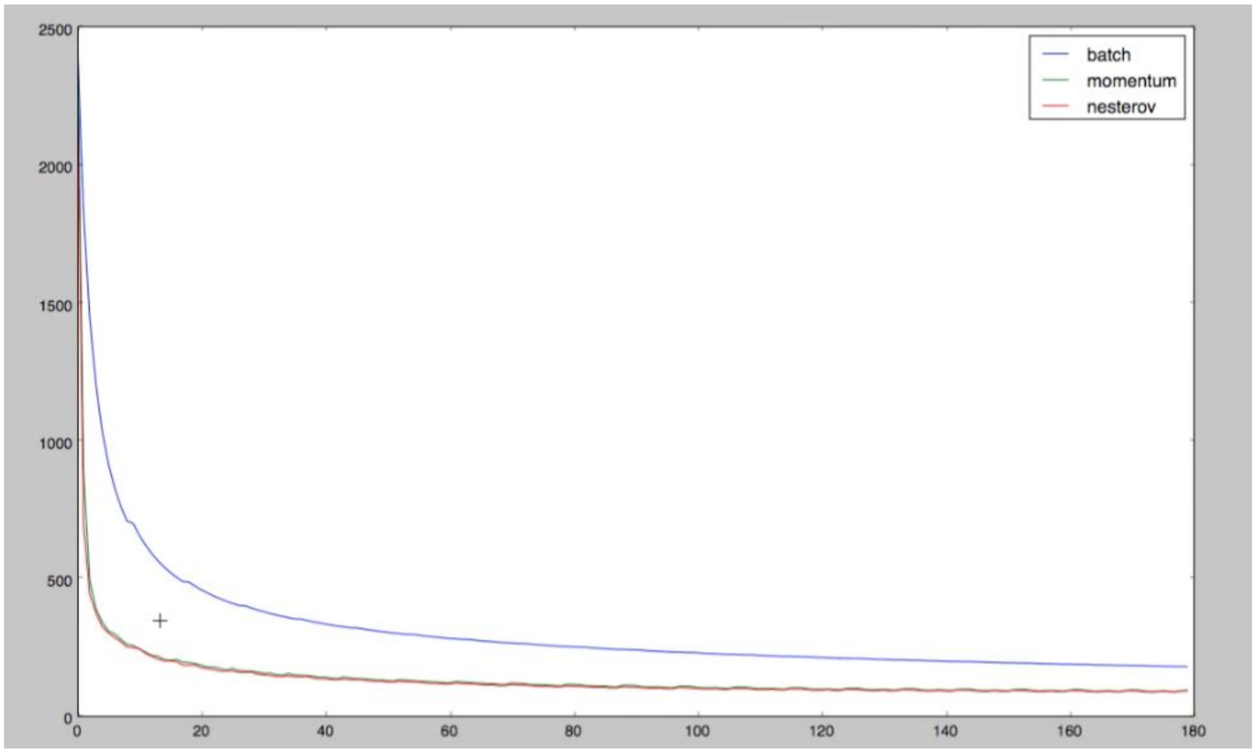
Nesterov momentum

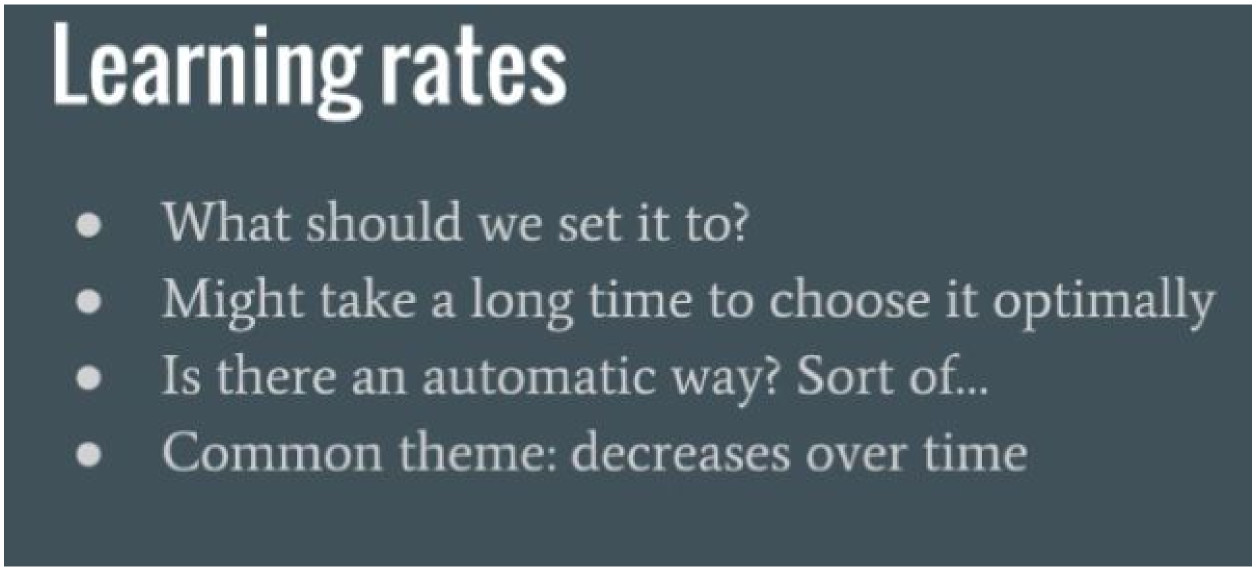
- Equivalent formulation:

$$v(t) = \mu v(t-1) - \eta \nabla J(t)$$

$$\Delta w(t) = -\mu v(t-1) + (1 + \mu) v(t)$$

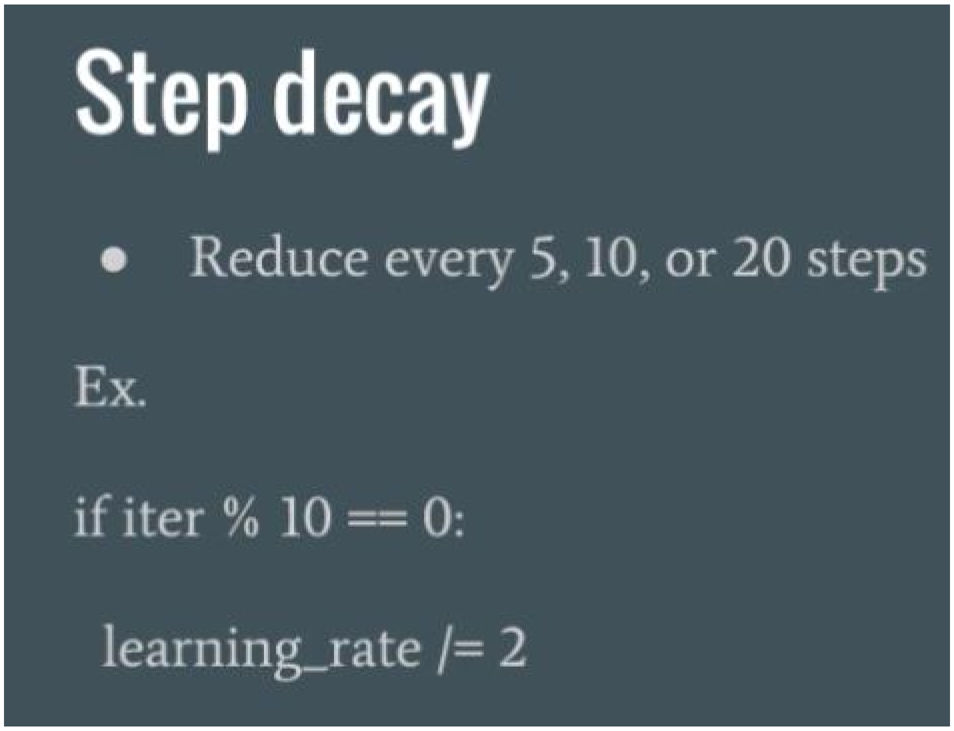
- This is the one we'll use





Learning rates

- What should we set it to?
- Might take a long time to choose it optimally
- Is there an automatic way? Sort of...
- Common theme: decreases over time

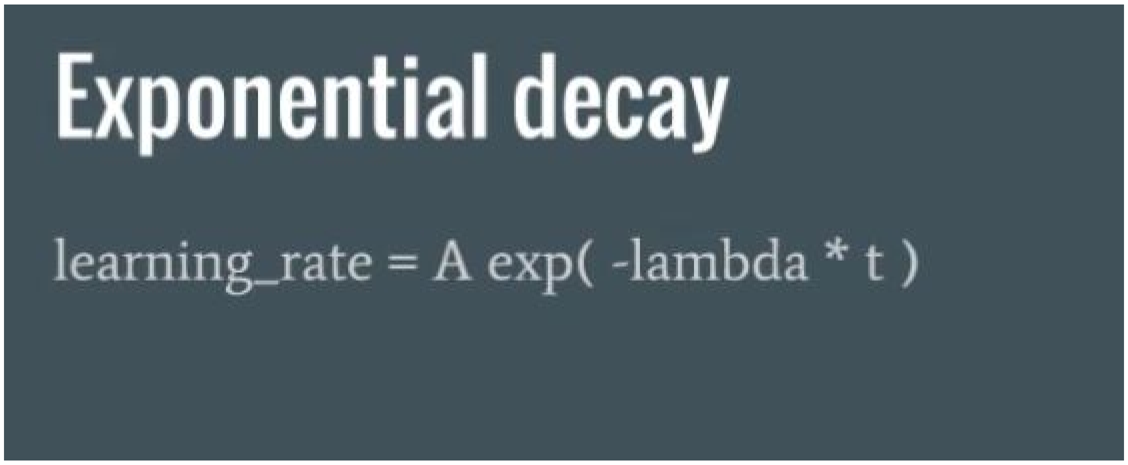


Step decay

- Reduce every 5, 10, or 20 steps

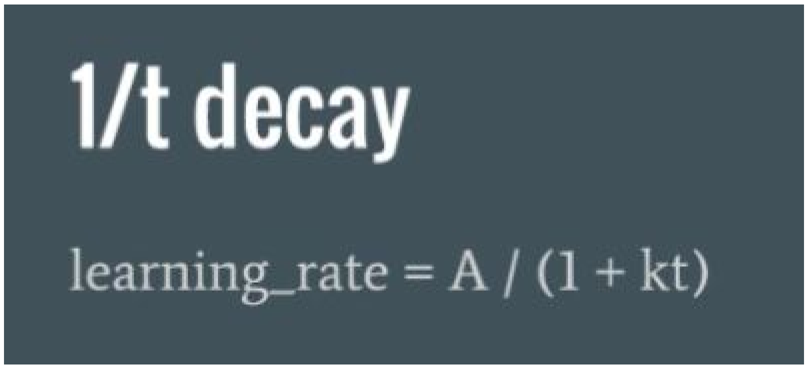
Ex.

```
if iter % 10 == 0:
  learning_rate /= 2
```



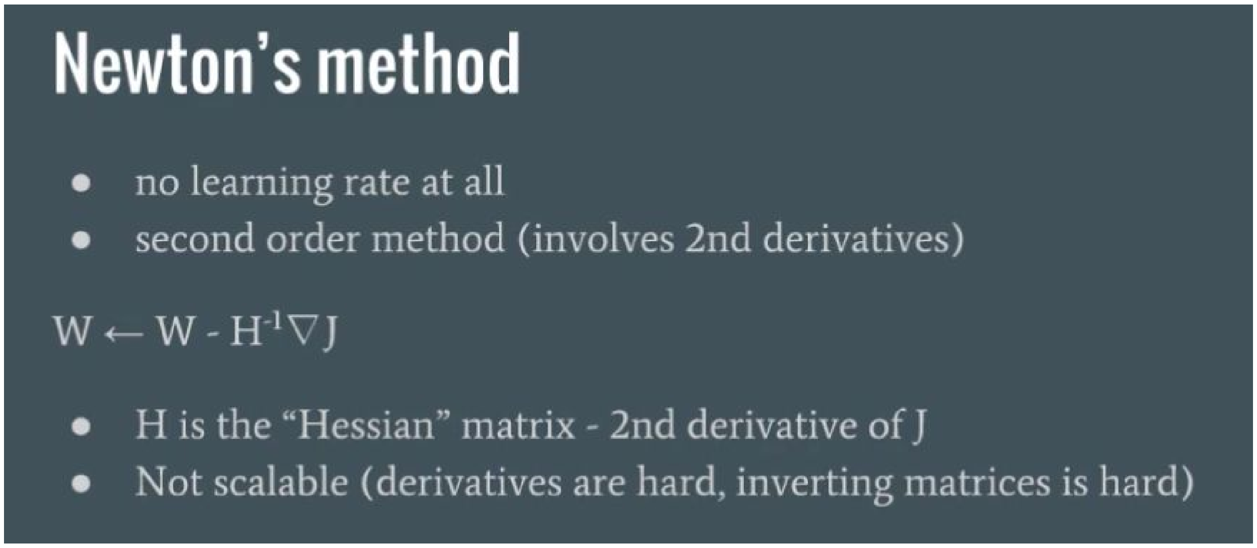
Exponential decay

learning_rate = A exp(-lambda * t)



1/t decay

learning_rate = A / (1 + kt)

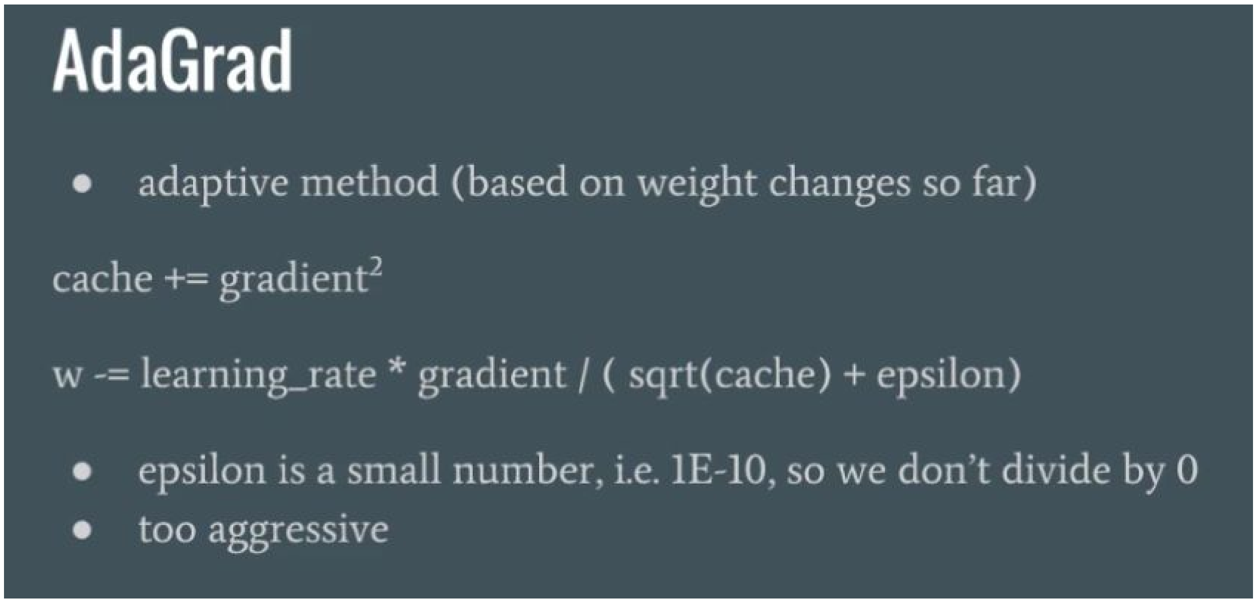


Newton's method

- no learning rate at all
- second order method (involves 2nd derivatives)

$W \leftarrow W - H^{-1}\nabla J$

- H is the "Hessian" matrix - 2nd derivative of J
- Not scalable (derivatives are hard, inverting matrices is hard)



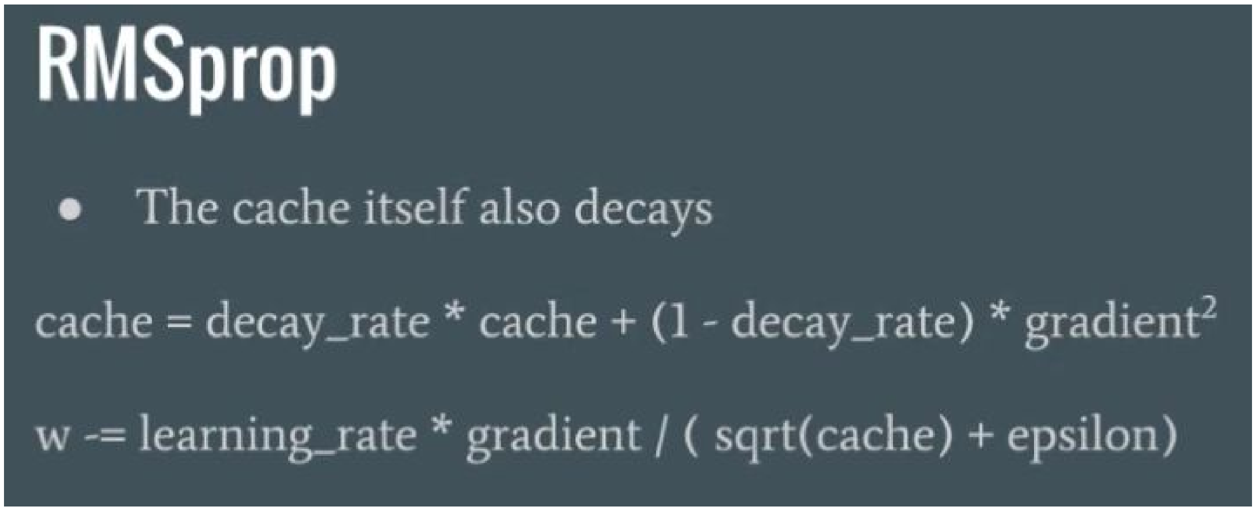
AdaGrad

- adaptive method (based on weight changes so far)

cache += gradient^2

w -= learning_rate * gradient / (sqrt(cache) + epsilon)

- epsilon is a small number, i.e. 1E-10, so we don't divide by 0
- too aggressive

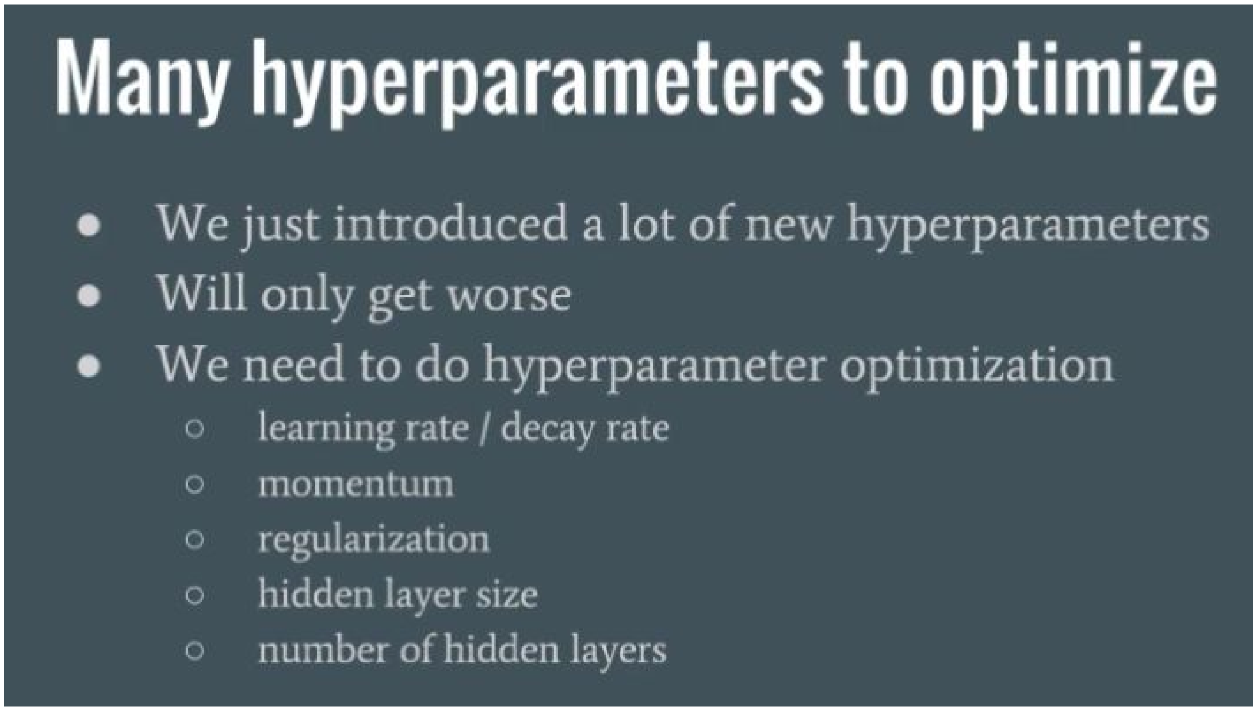


RMSprop

- The cache itself also decays

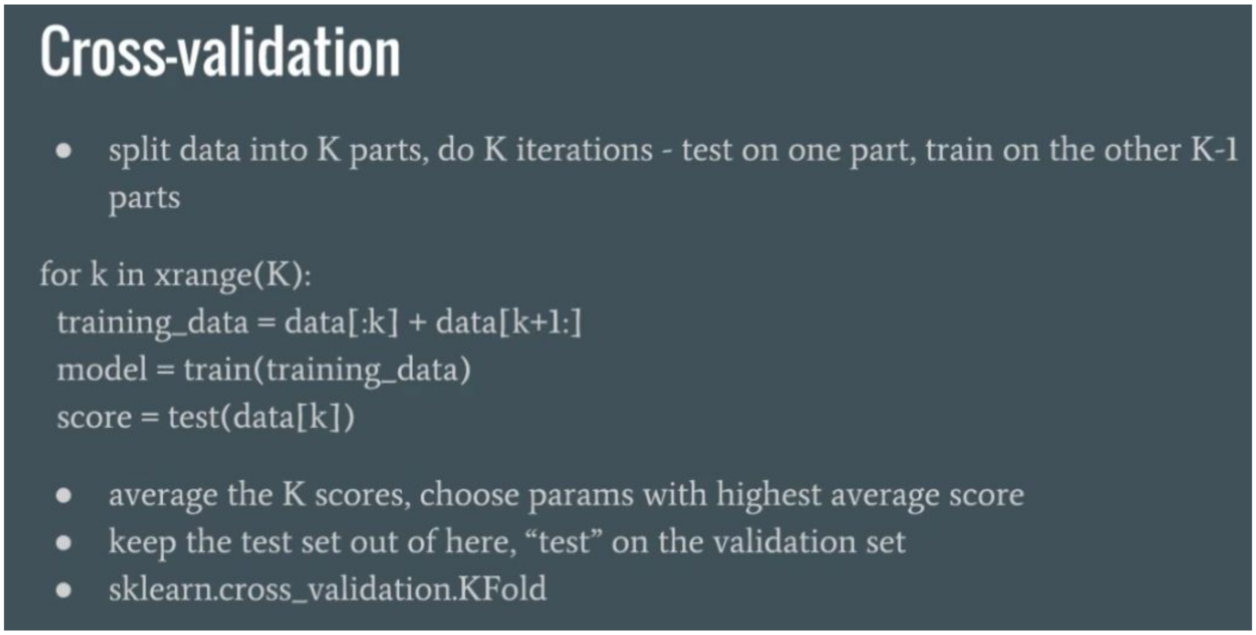
cache = decay_rate * cache + (1 - decay_rate) * gradient^2

w -= learning_rate * gradient / (sqrt(cache) + epsilon)



Many hyperparameters to optimize

- We just introduced a lot of new hyperparameters
- Will only get worse
- We need to do hyperparameter optimization
    - learning rate / decay rate
    - momentum
    - regularization
    - hidden layer size
    - number of hidden layers

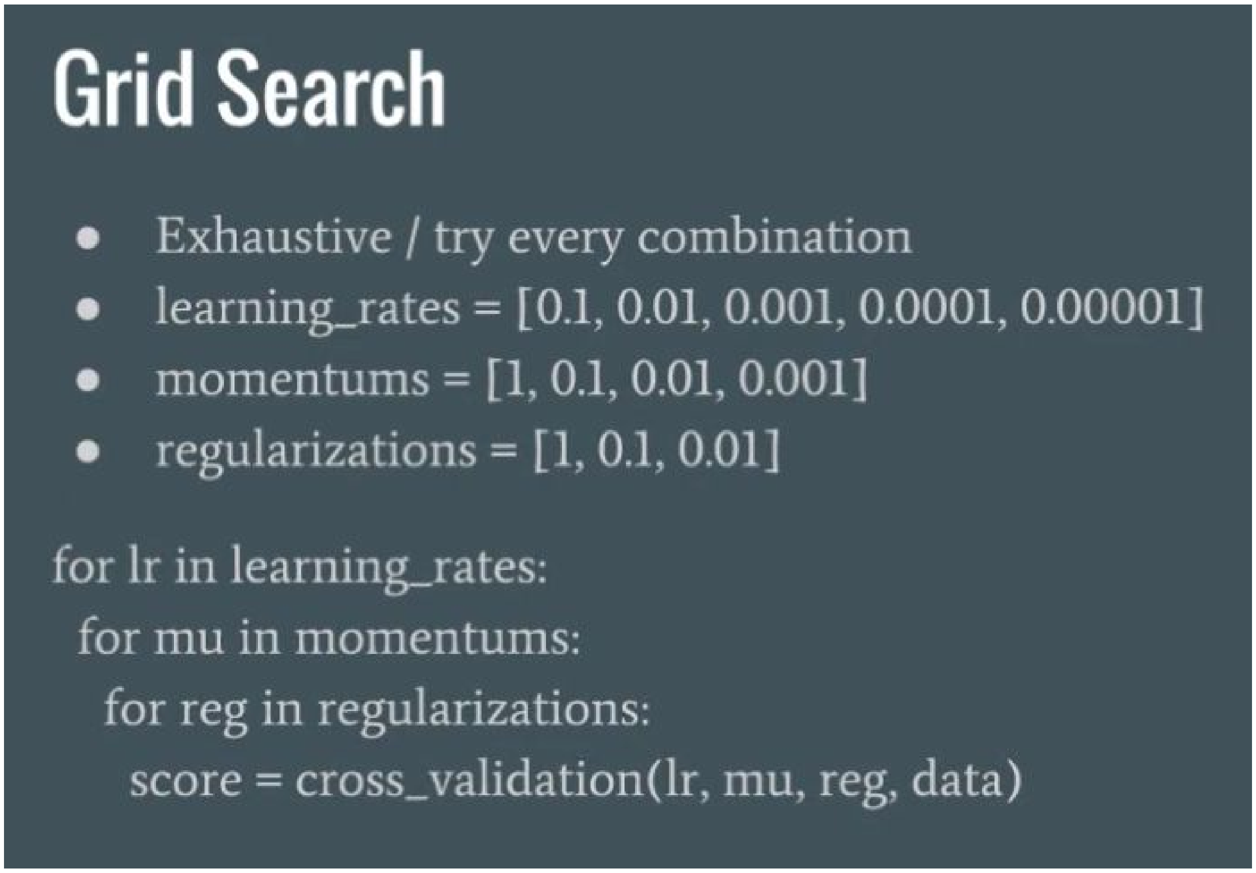


Cross-validation

- split data into K parts, do K iterations - test on one part, train on the other K-1 parts

```
for k in xrange(K):
 training_data = data[:k] + data[k+1:]
 model = train(training_data)
 score = test(data[k])
```

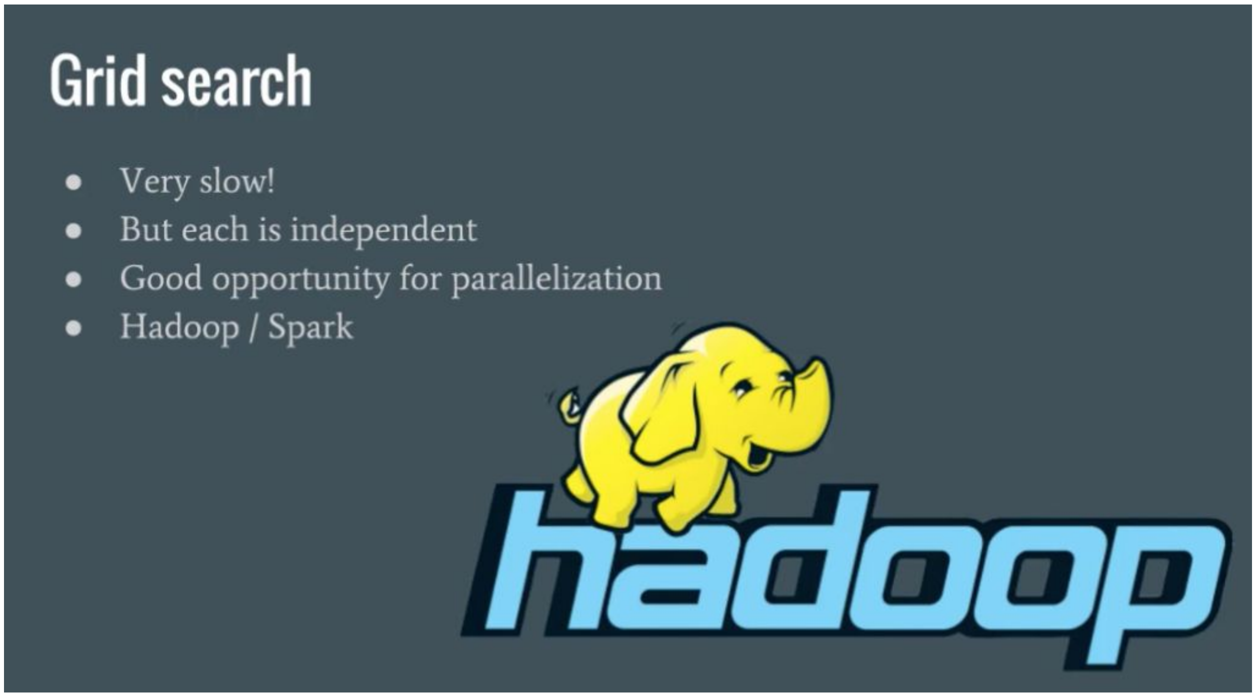
- average the K scores, choose params with highest average score
- keep the test set out of here, “test” on the validation set
- sklearn.cross_validation.KFold



Grid Search

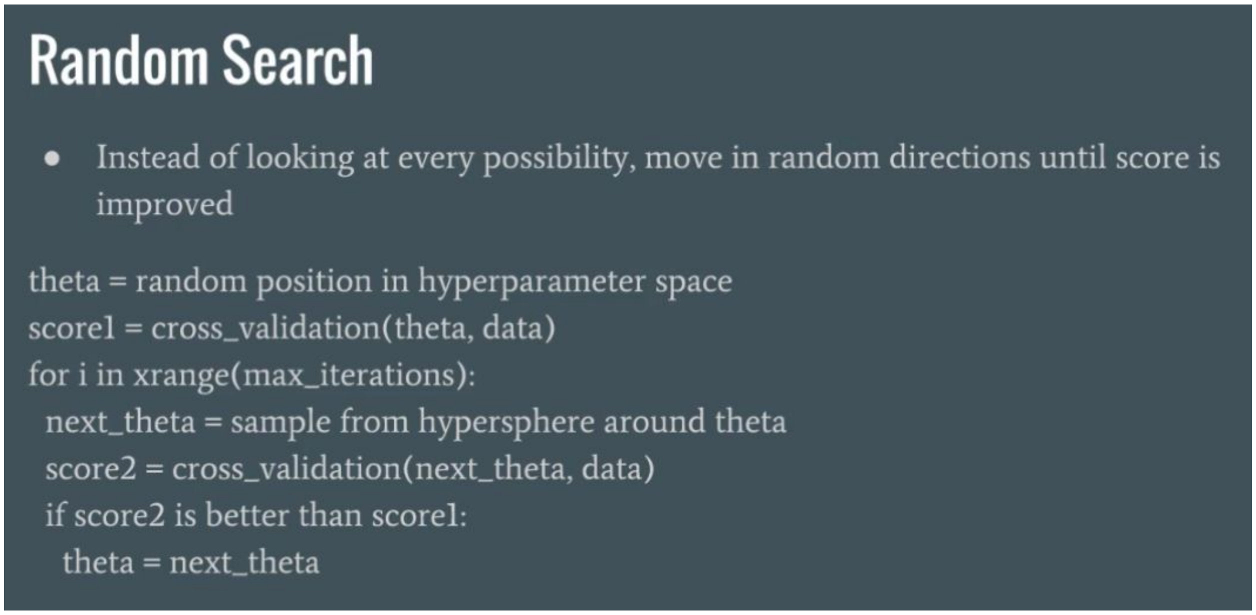
- Exhaustive / try every combination
- learning_rates = [0.1, 0.01, 0.001, 0.0001, 0.00001]
- momentums = [1, 0.1, 0.01, 0.001]
- regularizations = [1, 0.1, 0.01]

```
for lr in learning_rates:
  for mu in momentums:
    for reg in regularizations:
      score = cross_validation(lr, mu, reg, data)
```



Grid search

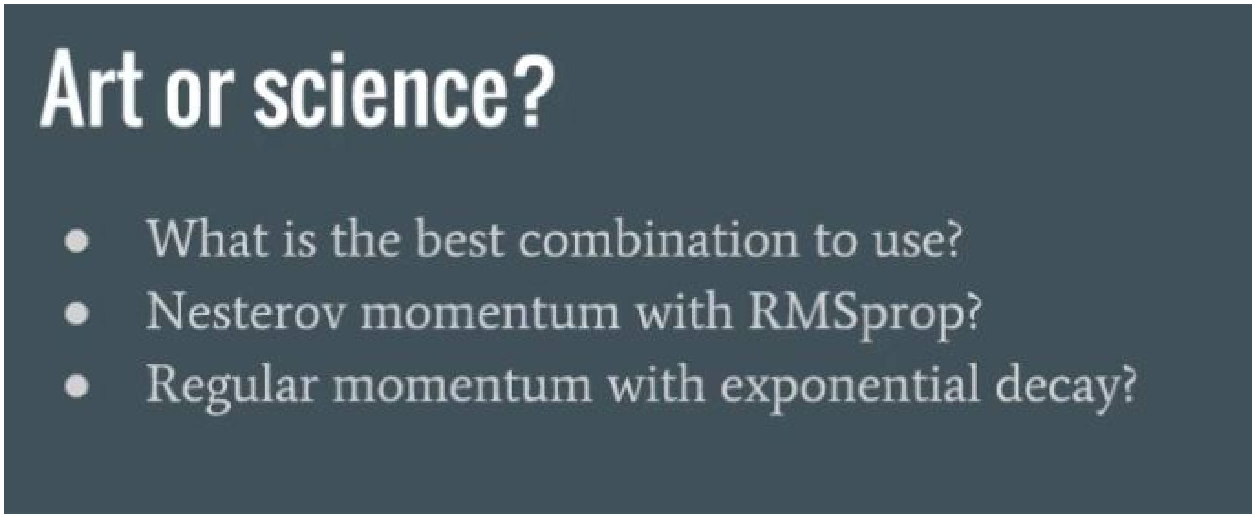
- Very slow!
- But each is independent
- Good opportunity for parallelization
- Hadoop / Spark



Random Search

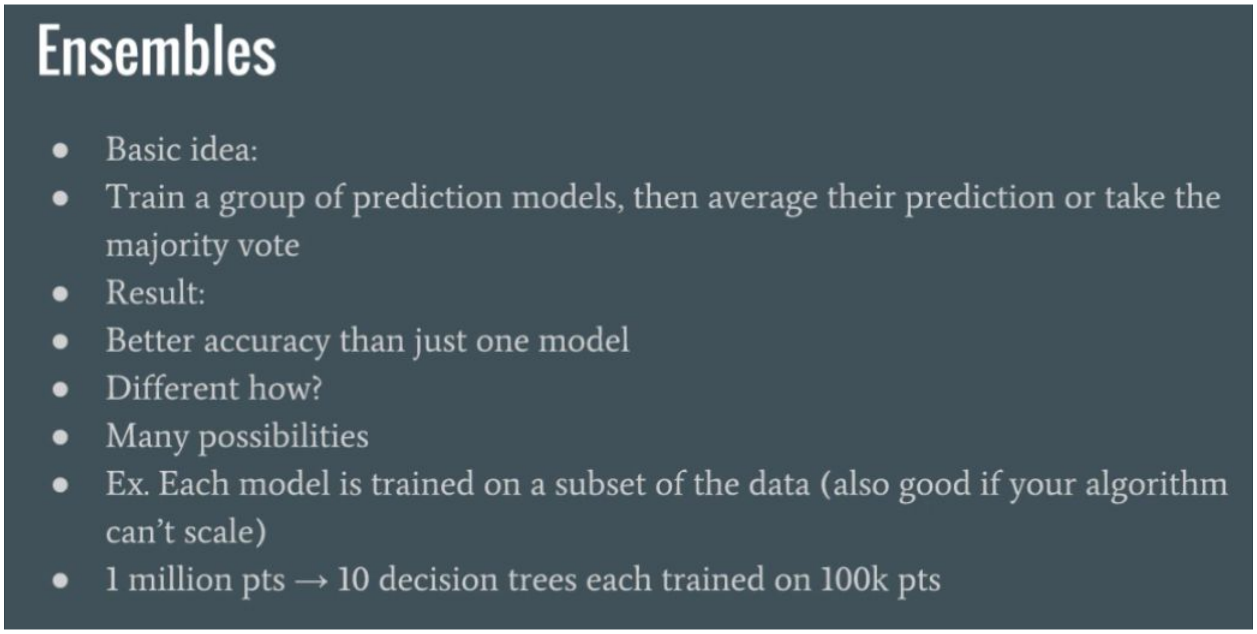
- Instead of looking at every possibility, move in random directions until score is improved

```
theta = random position in hyperparameter space
score1 = cross_validation(theta, data)
for i in xrange(max_iterations):
  next_theta = sample from hypersphere around theta
  score2 = cross_validation(next_theta, data)
  if score2 is better than score1:
    theta = next_theta
```



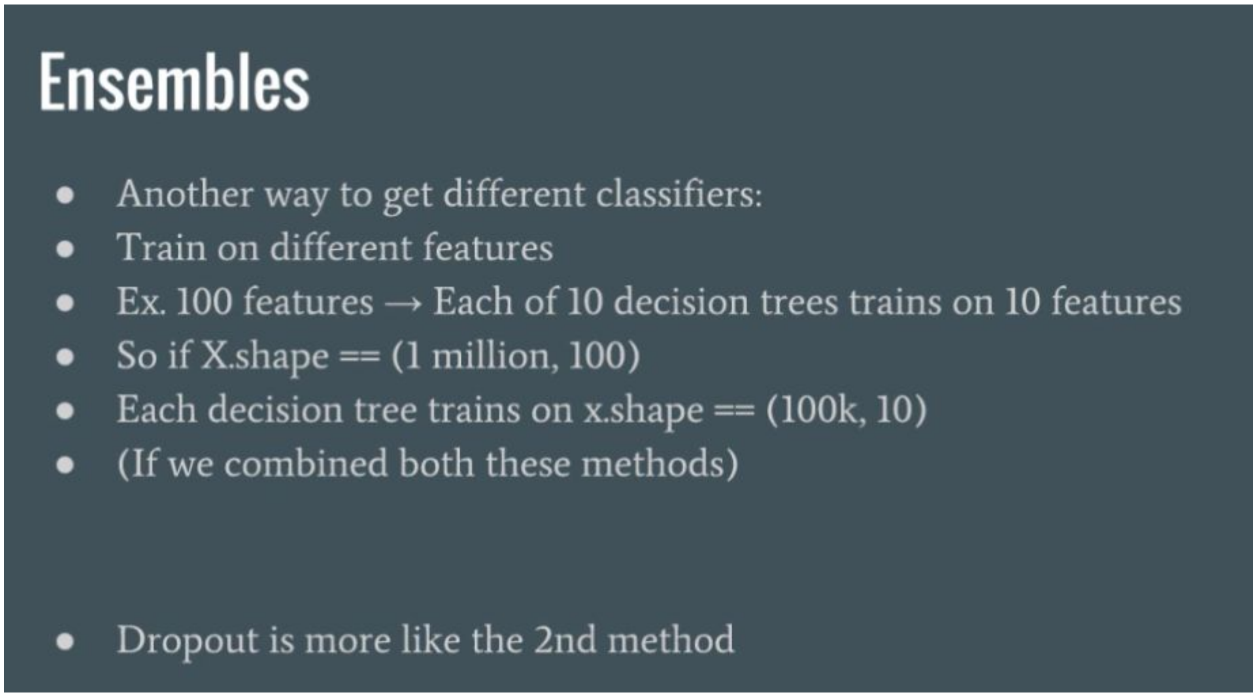
Art or science?

- What is the best combination to use?
- Nesterov momentum with RMSprop?
- Regular momentum with exponential decay?



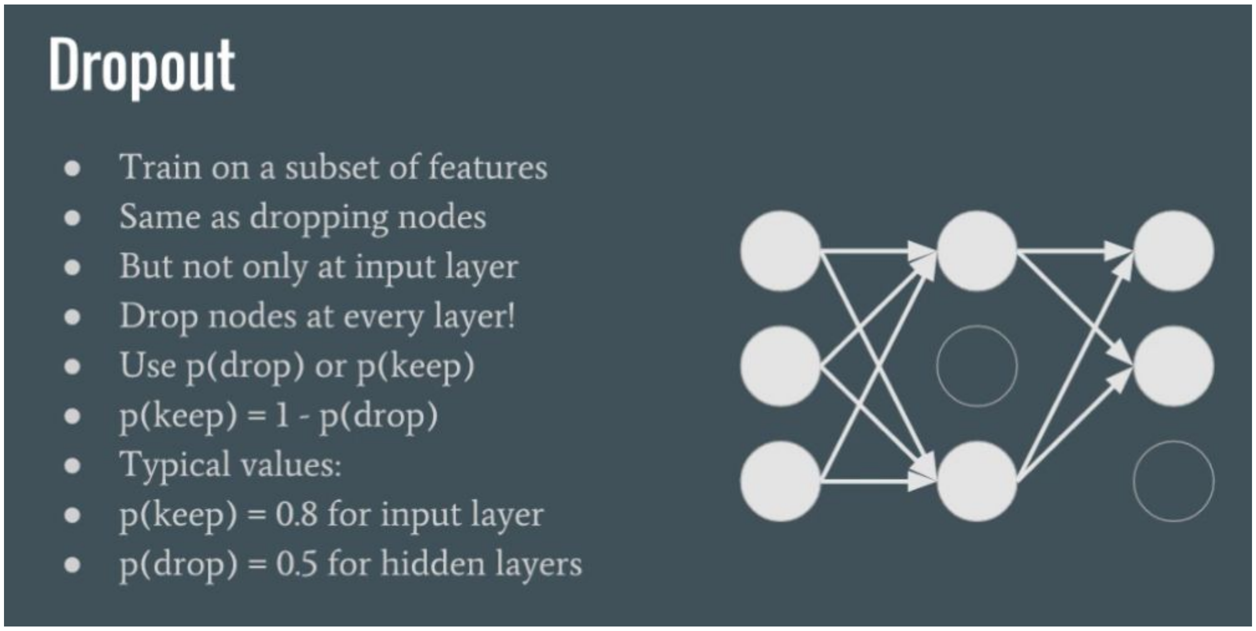
Ensembles

- Basic idea:
- Train a group of prediction models, then average their prediction or take the majority vote
- Result:
- Better accuracy than just one model
- Different how?
- Many possibilities
- Ex. Each model is trained on a subset of the data (also good if your algorithm can't scale)
- 1 million pts → 10 decision trees each trained on 100k pts



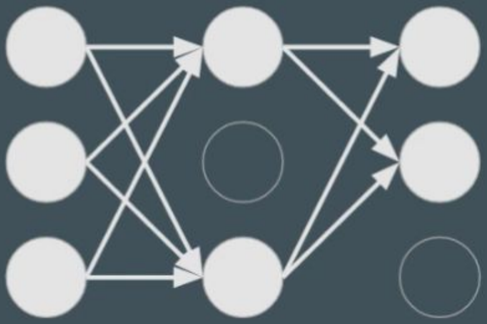
Ensembles

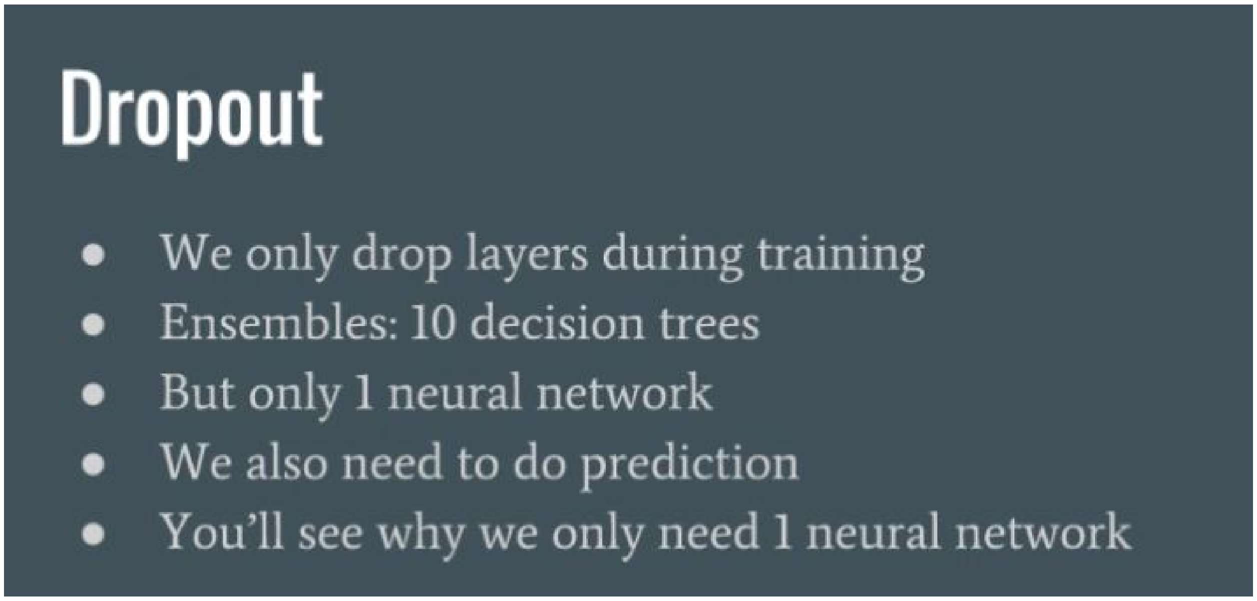
- Another way to get different classifiers:
- Train on different features
- Ex. 100 features → Each of 10 decision trees trains on 10 features
- So if X.shape == (1 million, 100)
- Each decision tree trains on x.shape == (100k, 10)
- (If we combined both these methods)
- Dropout is more like the 2nd method



Dropout

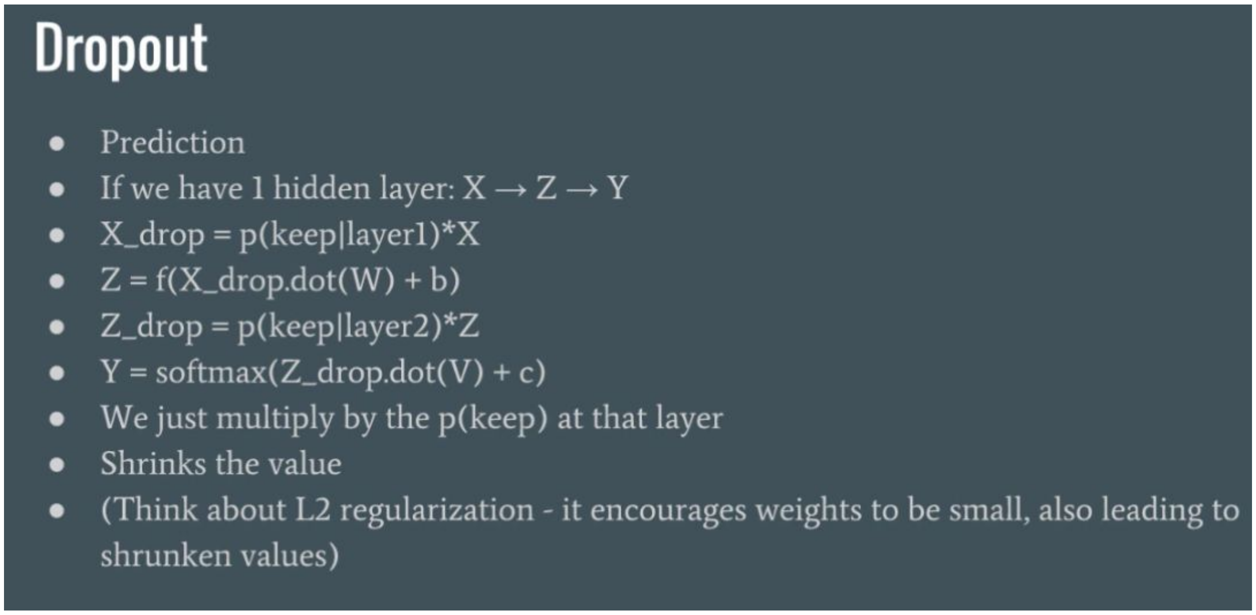
- Train on a subset of features
- Same as dropping nodes
- But not only at input layer
- Drop nodes at every layer!
- Use p(drop) or p(keep)
- p(keep) = 1 - p(drop)
- Typical values:
- p(keep) = 0.8 for input layer
- p(drop) = 0.5 for hidden layers





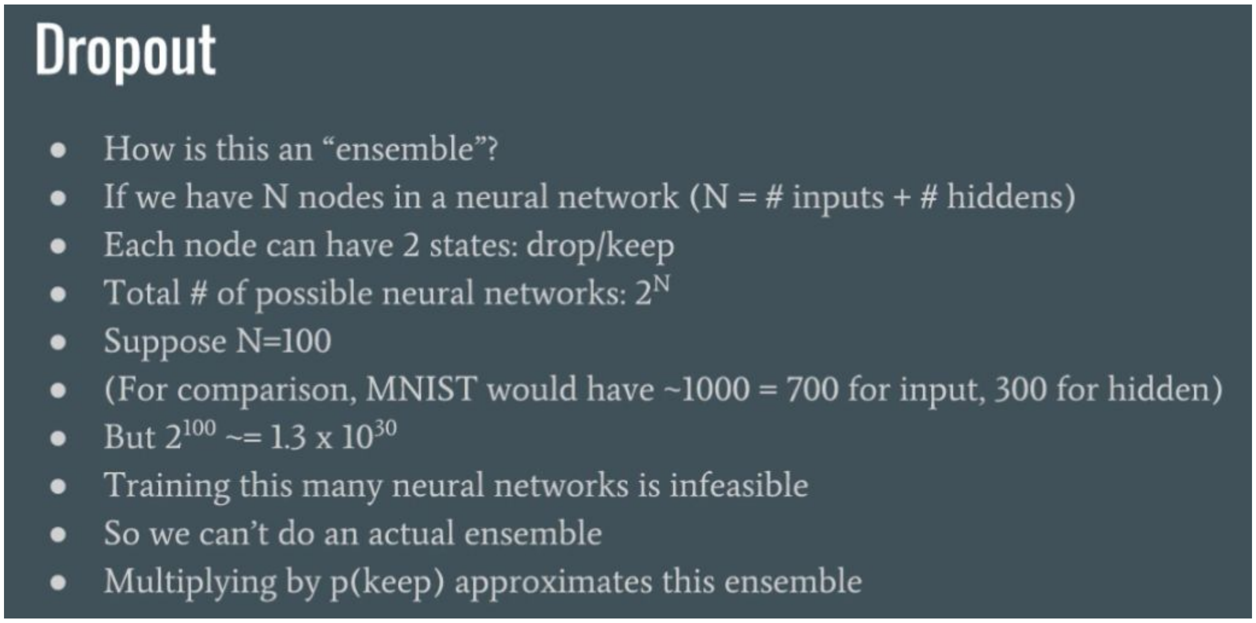
Dropout

- We only drop layers during training
- Ensembles: 10 decision trees
- But only 1 neural network
- We also need to do prediction
- You’ll see why we only need 1 neural network



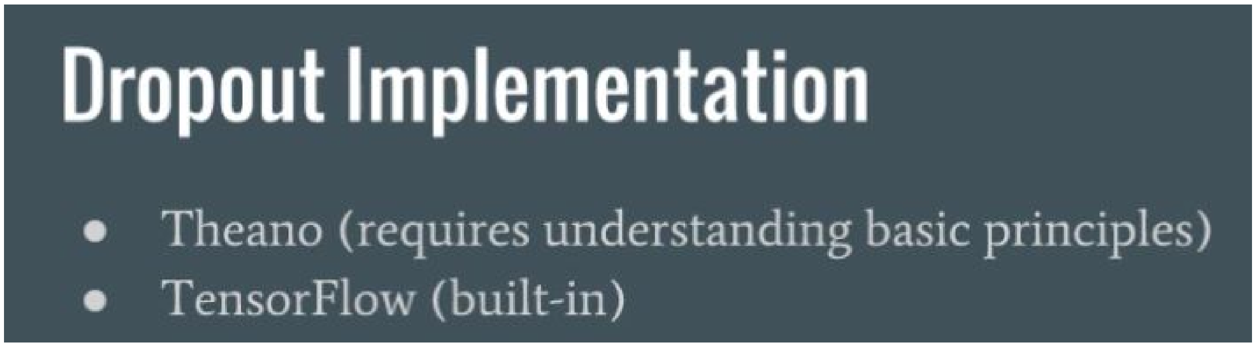
Dropout

- Prediction
- If we have 1 hidden layer: $X \rightarrow Z \rightarrow Y$
- X_drop = p(keep|layer1)*X
- Z = f(X_drop.dot(W) + b)
- Z_drop = p(keep|layer2)*Z
- Y = softmax(Z_drop.dot(V) + c)
- We just multiply by the p(keep) at that layer
- Shrinks the value
- (Think about L2 regularization - it encourages weights to be small, also leading to shrunken values)



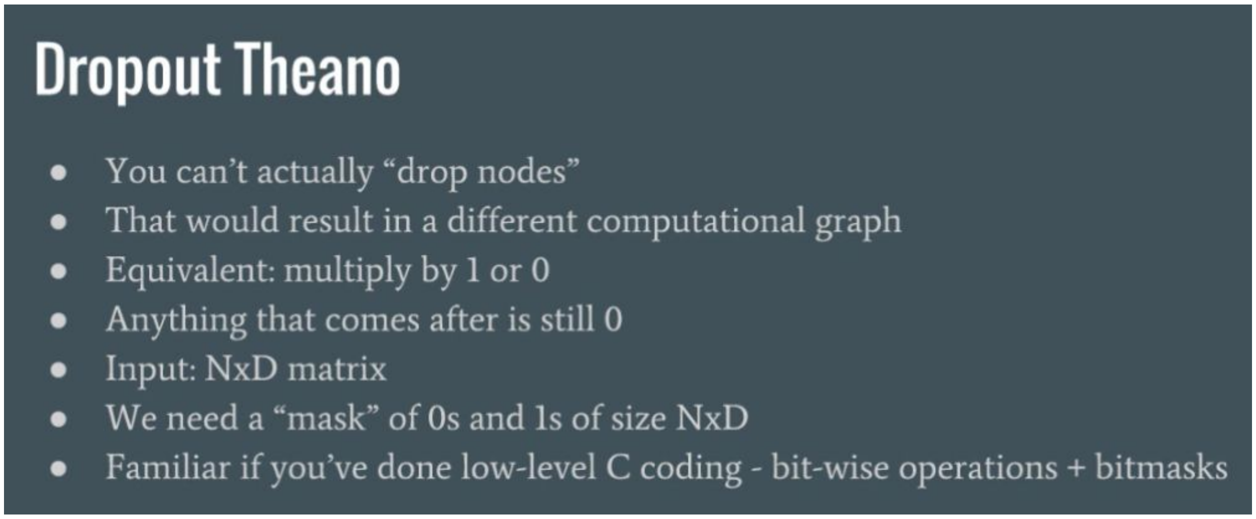
Dropout

- How is this an “ensemble”?
- If we have N nodes in a neural network (N = # inputs + # hiddens)
- Each node can have 2 states: drop/keep
- Total # of possible neural networks: 2^N
- Suppose N=100
- (For comparison, MNIST would have ~1000 = 700 for input, 300 for hidden)
- But 2^{100} ~= 1.3 x 10^{30}
- Training this many neural networks is infeasible
- So we can’t do an actual ensemble
- Multiplying by p(keep) approximates this ensemble



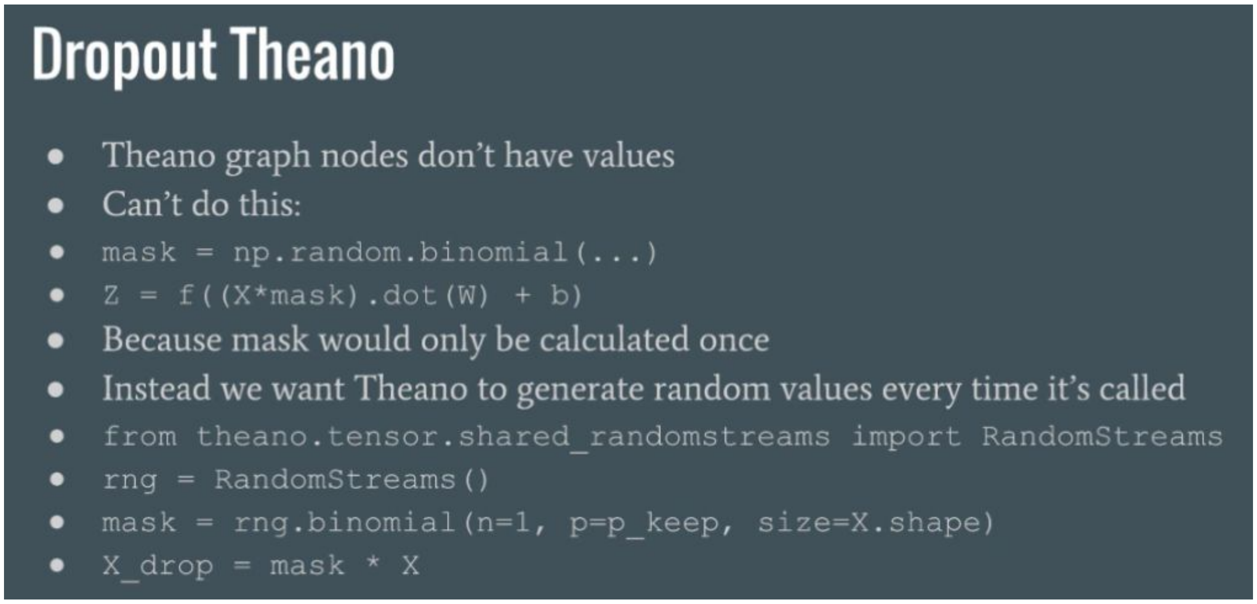
Dropout Implementation

- Theano (requires understanding basic principles)
- TensorFlow (built-in)



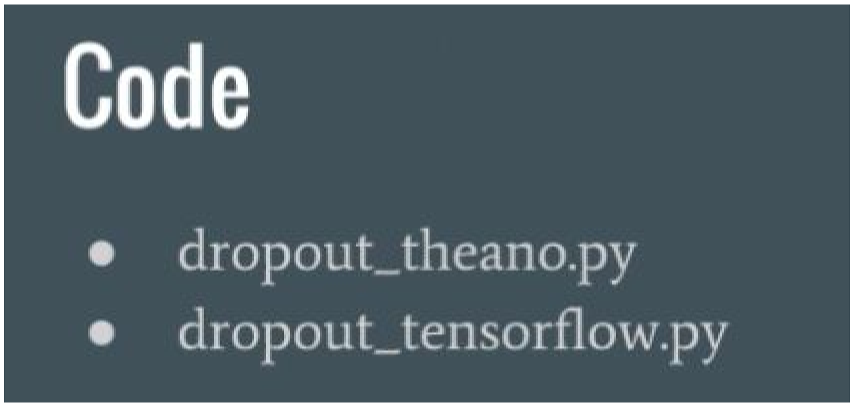
Dropout Theano

- You can’t actually “drop nodes”
- That would result in a different computational graph
- Equivalent: multiply by 1 or 0
- Anything that comes after is still 0
- Input: NxD matrix
- We need a “mask” of 0s and 1s of size NxD
- Familiar if you’ve done low-level C coding - bit-wise operations + bitmasks



Dropout Theano

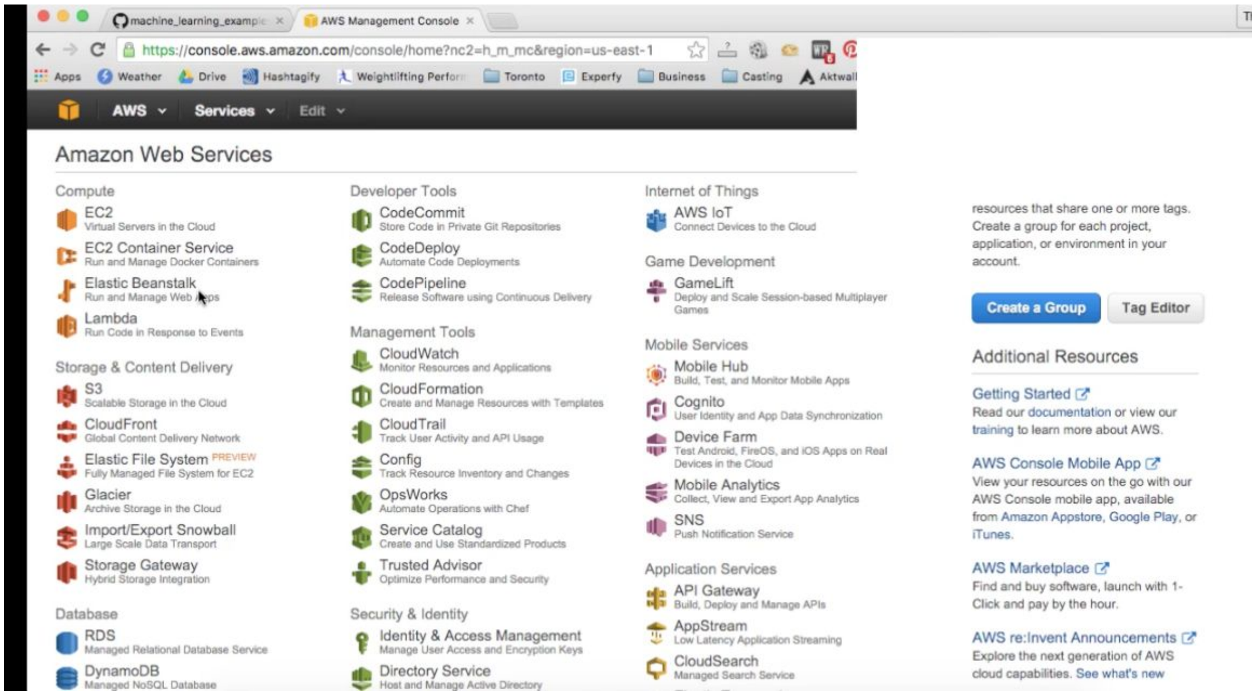
- Theano graph nodes don’t have values
- Can’t do this:
- `mask = np.random.binomial(...)`
- `Z = f((X*mask).dot(W) + b)`
- Because mask would only be calculated once
- Instead we want Theano to generate random values every time it’s called
- `from theano.tensor.shared_randomstreams import RandomStreams`
- `rng = RandomStreams()`
- `mask = rng.binomial(n=1, p=p_keep, size=X.shape)`
- `X_drop = mask * X`



Code

- dropout_theano.py
- dropout_tensorflow.py

SETTING UP THE GPU INSTANCES FOR THE AMAZON AWS (IN MAC OS)



Amazon Web Services

Compute
EC2 Virtual Servers in the Cloud
EC2 Container Service Run and Manage Docker Containers
Elastic Beanstalk Run and Manage Web Apps
Lambda Run Code in Response to Events

Storage & Content Delivery
S3 Scalable Storage in the Cloud
CloudFront Global Content Delivery Network
Elastic File System PREVIEW Fully Managed File System for EC2
Glacier Archive Storage in the Cloud
Import/Export Snowball Large Scale Data Transport
Storage Gateway Hybrid Storage Integration

Database
RDS Managed Relational Database Service
DynamoDB Managed NoSQL Database

Developer Tools
CodeCommit Store Code in Private Git Repositories
CodeDeploy Automate Code Deployments
CodePipeline Release Software using Continuous Delivery

Management Tools
CloudWatch Monitor Resources and Applications
CloudFormation Create and Manage Resources with Templates
CloudTrail Track User Activity and API Usage
Config Track Resource Inventory and Changes
OpsWorks Automate Operations with Chef
Service Catalog Create and Use Standardized Products
Trusted Advisor Optimize Performance and Security

Security & Identity
Identity & Access Management Manage User Access and Encryption Keys
Directory Service Host and Manage Active Directory

Internet of Things
AWS IoT Connect Devices to the Cloud

Game Development
GameLift Deploy and Scale Session-based Multiplayer Games

Mobile Services
Mobile Hub Build, Test, and Monitor Mobile Apps
Cognito User Identity and App Data Synchronization
Device Farm Test Android, FireOS, and iOS Apps on Real Devices in the Cloud
Mobile Analytics Collect, View and Export App Analytics
SNS Push Notification Service

Application Services
API Gateway Build, Deploy and Manage APIs
AppStream Low Latency Application Streaming
CloudSearch Managed Search Service

resources that share one or more tags. Create a group for each project, application, or environment in your account.

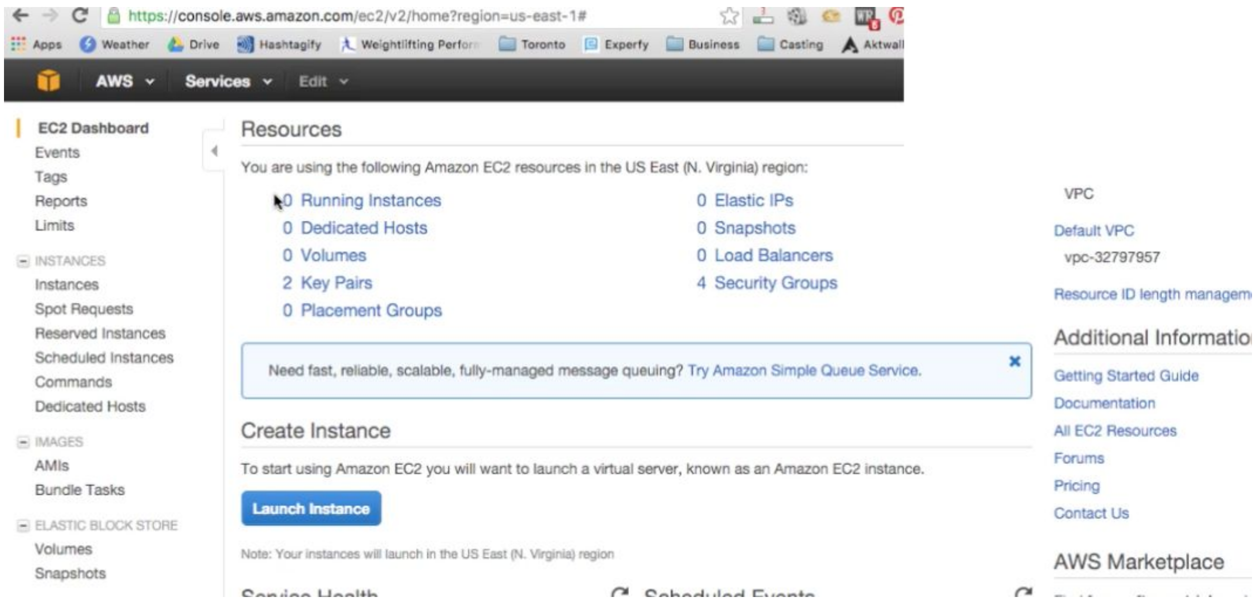
Additional Resources

Getting Started
Read our documentation or view our training to learn more about AWS.

AWS Console Mobile App
View your resources on the go with our AWS Console mobile app, available from Amazon Appstore, Google Play, or iTunes.

AWS Marketplace
Find and buy software, launch with 1-Click and pay by the hour.

AWS re:Invent Announcements
Explore the next generation of AWS cloud capabilities. See what's new



Resources

You are using the following Amazon EC2 resources in the US East (N. Virginia) region:

0 Running Instances	0 Elastic IPs
0 Dedicated Hosts	0 Snapshots
0 Volumes	0 Load Balancers
2 Key Pairs	4 Security Groups
0 Placement Groups	

Need fast, reliable, scalable, fully-managed message queuing? Try Amazon Simple Queue Service.

Create Instance

To start using Amazon EC2 you will want to launch a virtual server, known as an Amazon EC2 instance.

Note: Your instances will launch in the US East (N. Virginia) region

Service Health

Scheduled Events

VPC

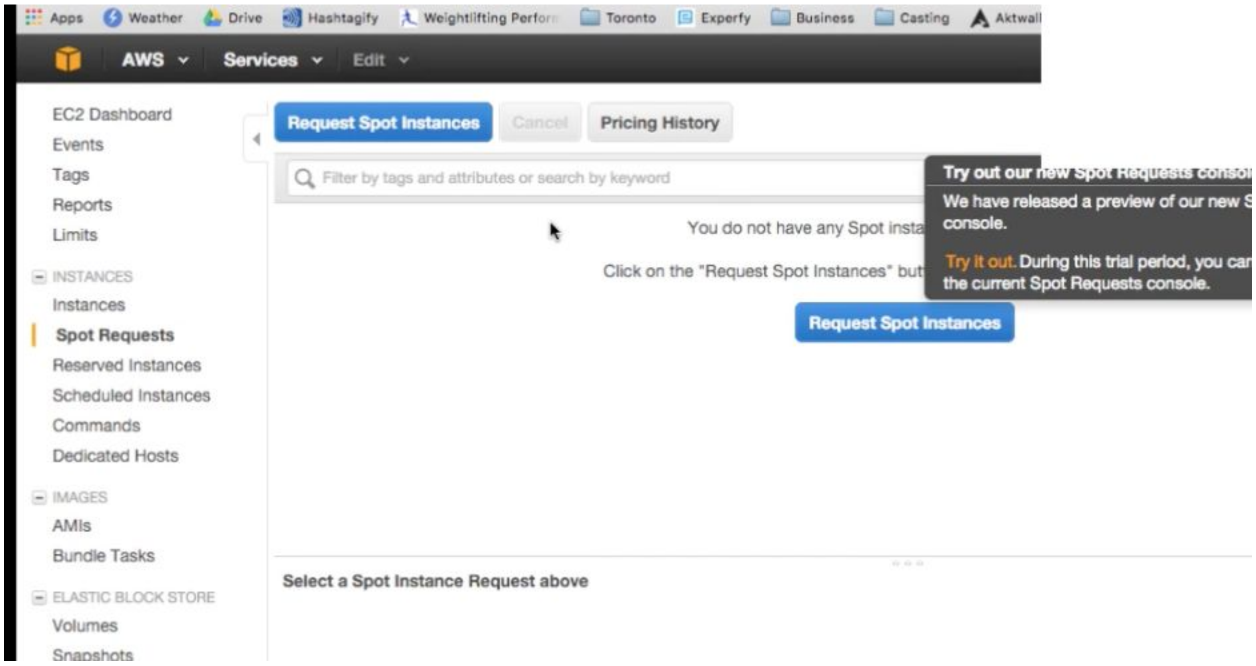
Default VPC
vpc-32797957

Resource ID length managem

Additional Informatio

Getting Started Guide
Documentation
All EC2 Resources
Forums
Pricing
Contact Us

AWS Marketplace



You do not have any Spot insta

Click on the "Request Spot Instances" but

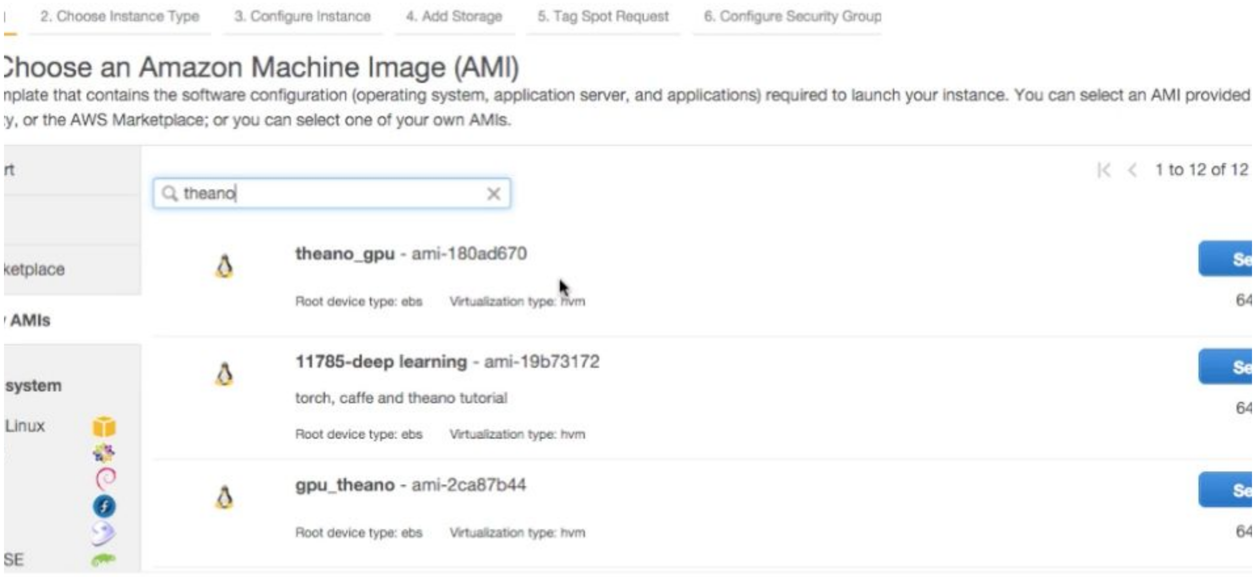
Try out our new Spot Requests consol

We have released a preview of our new S console.

Try it out. During this trial period, you car the current Spot Requests console.

Select a Spot Instance Request above

Need to choose **gpu_theano** for the community AMI.



Choose an Amazon Machine Image (AMI)

nplate that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided y, or the AWS Marketplace; or you can select one of your own AMIs.

theano_gpu - ami-180ad670

Root device type: ebs Virtualization type: hvm

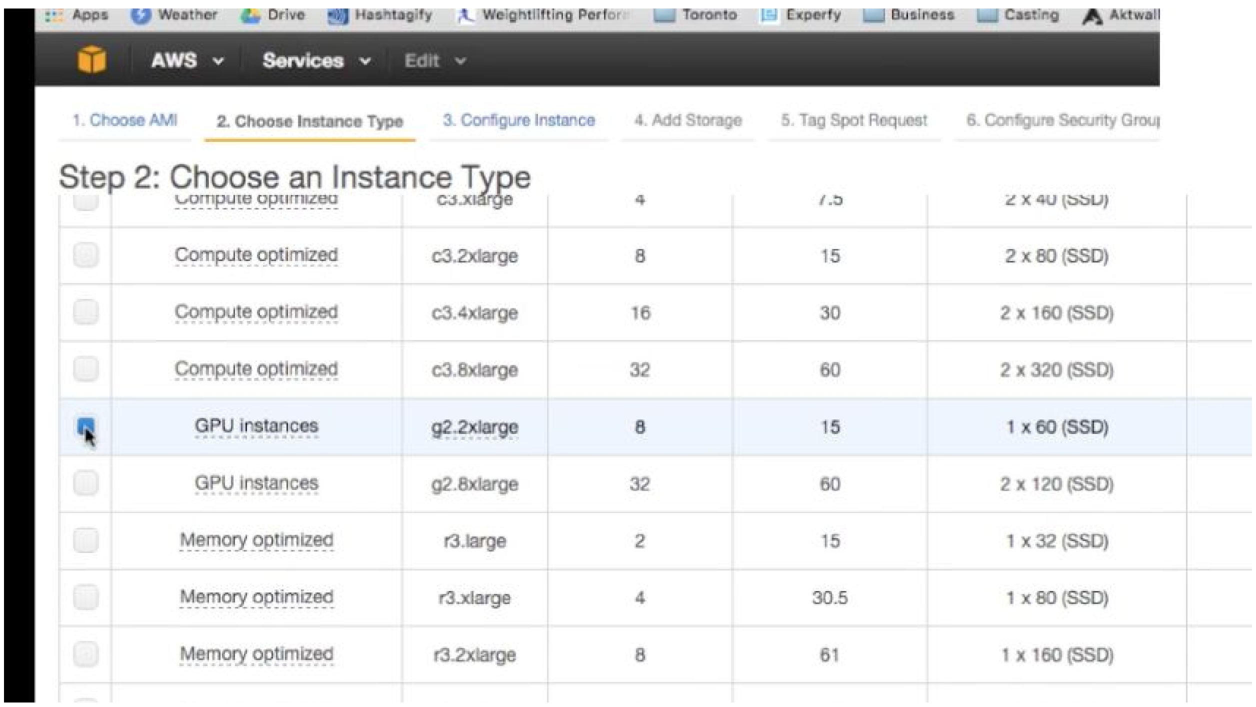
11785-deep learning - ami-19b73172

torch, caffe and theano tutorial

Root device type: ebs Virtualization type: hvm

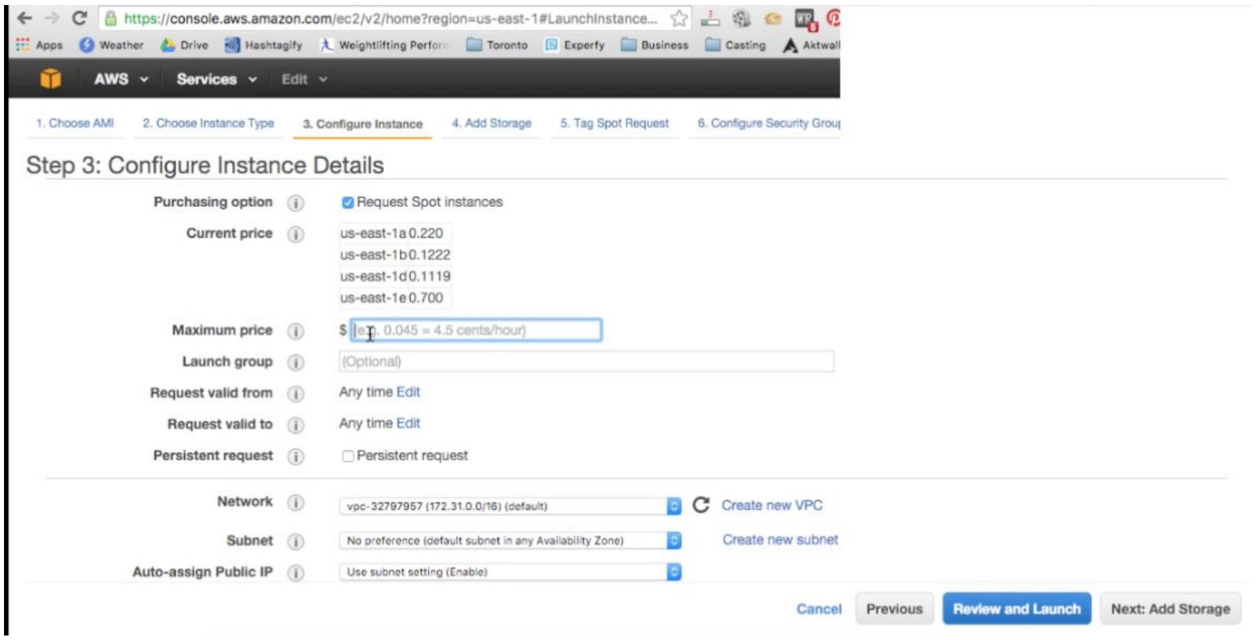
gpu_theano - ami-2ca87b44

Root device type: ebs Virtualization type: hvm



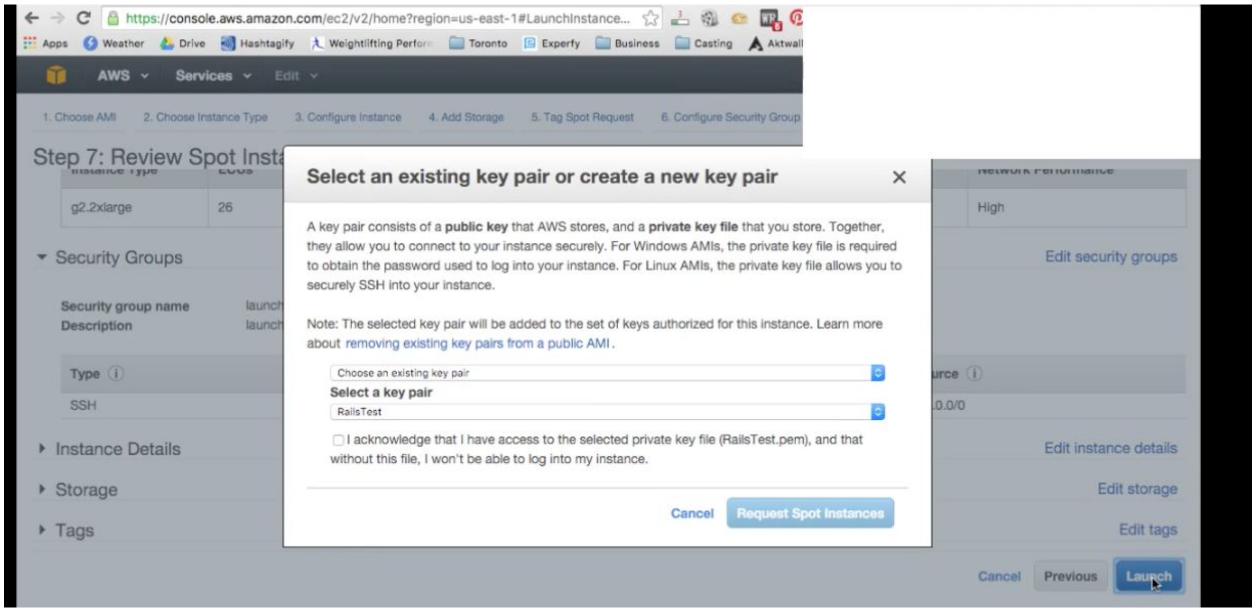
Step 2: Choose an Instance Type

☐	Compute optimized	c3.xlarge	4	7.5	2 x 40 (SSD)
☐	Compute optimized	c3.2xlarge	8	15	2 x 80 (SSD)
☐	Compute optimized	c3.4xlarge	16	30	2 x 160 (SSD)
☐	Compute optimized	c3.8xlarge	32	60	2 x 320 (SSD)
☑	GPU instances	g2.2xlarge	8	15	1 x 60 (SSD)
☐	GPU instances	g2.8xlarge	32	60	2 x 120 (SSD)
☐	Memory optimized	r3.large	2	15	1 x 32 (SSD)
☐	Memory optimized	r3.xlarge	4	30.5	1 x 80 (SSD)
☐	Memory optimized	r3.2xlarge	8	61	1 x 160 (SSD)



Step 3: Configure Instance Details

Purchasing option	☑ Request Spot instances
Current price	us-east-1a 0.220 us-east-1b 0.1222 us-east-1d 0.1119 us-east-1e 0.700
Maximum price	$ (e.g. 0.045 = 4.5 cents/hour)
Launch group	(Optional)
Request valid from	Any time Edit
Request valid to	Any time Edit
Persistent request	☐ Persistent request
Network	vpc-32797957 (172.31.0.0/16) (default) — Create new VPC
Subnet	No preference (default subnet in any Availability Zone) — Create new subnet
Auto-assign Public IP	Use subnet setting (Enable)



Select an existing key pair or create a new key pair

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about removing existing key pairs from a public AMI.

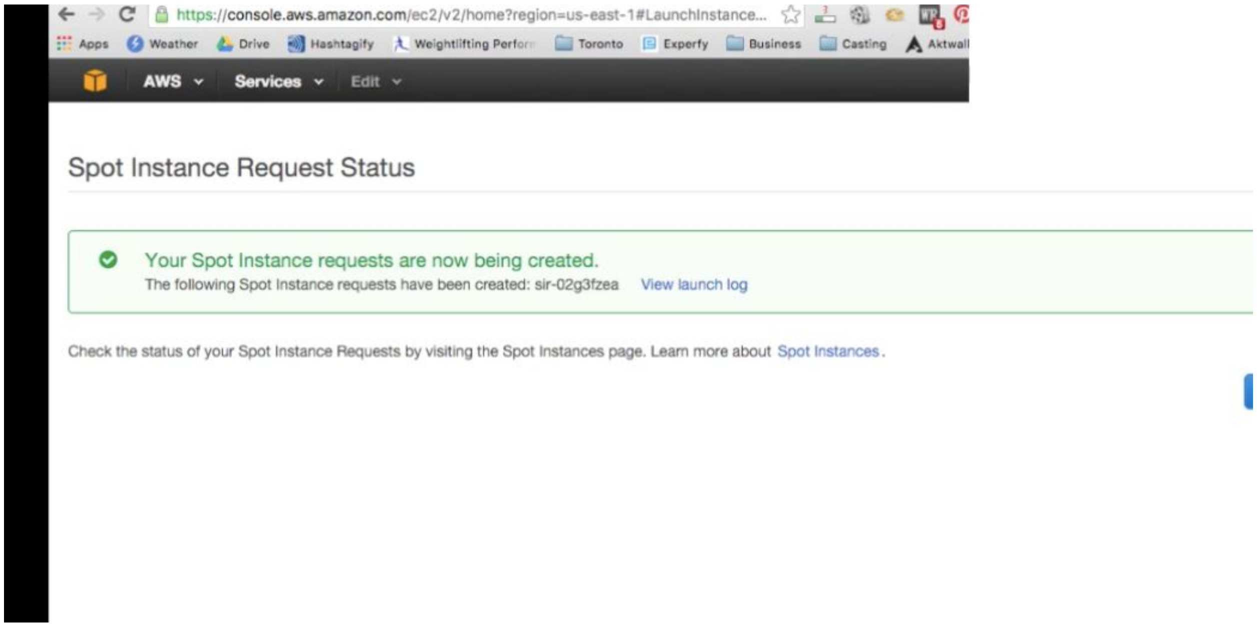
Choose an existing key pair

Select a key pair

RailsTest

☐ I acknowledge that I have access to the selected private key file (RailsTest.pem), and that without this file, I won't be able to log into my instance.

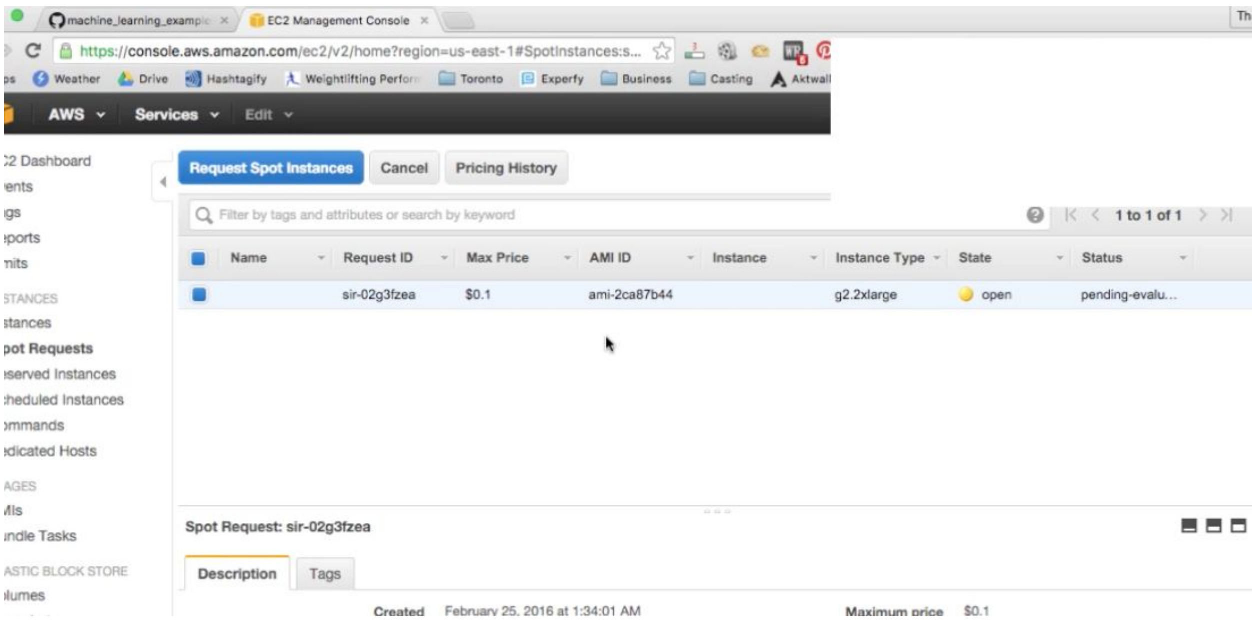
Cancel | Request Spot Instances


https://console.aws.amazon.com/ec2/v2/home?region=us-east-1#LaunchInstance...
AWS
Services
Edit

Spot Instance Request Status

Your Spot Instance requests are now being created.
The following Spot Instance requests have been created: sir-02g3fzea View launch log

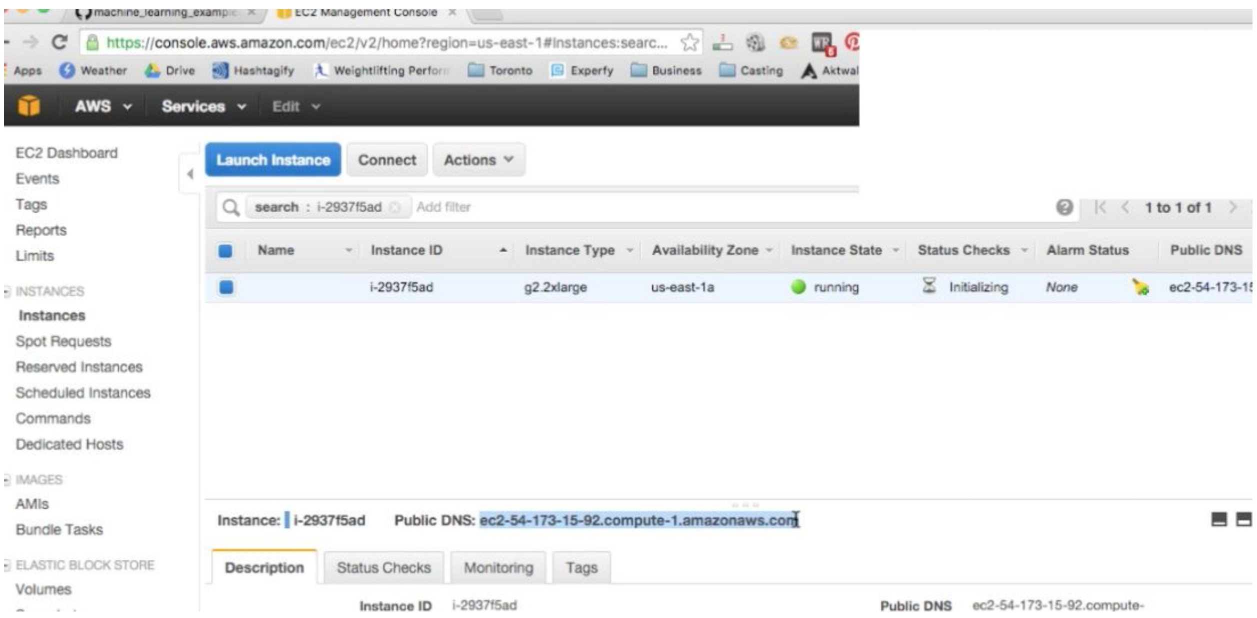
Check the status of your Spot Instance Requests by visiting the Spot Instances page. Learn more about Spot Instances.



Name	Request ID	Max Price	AMI ID	Instance	Instance Type	State	Status
	sir-02g3fzea	$0.1	ami-2ca87b44		g2.2xlarge	open	pending-evalu...

Spot Request: sir-02g3fzea

Created February 25, 2016 at 1:34:01 AM　Maximum price $0.1



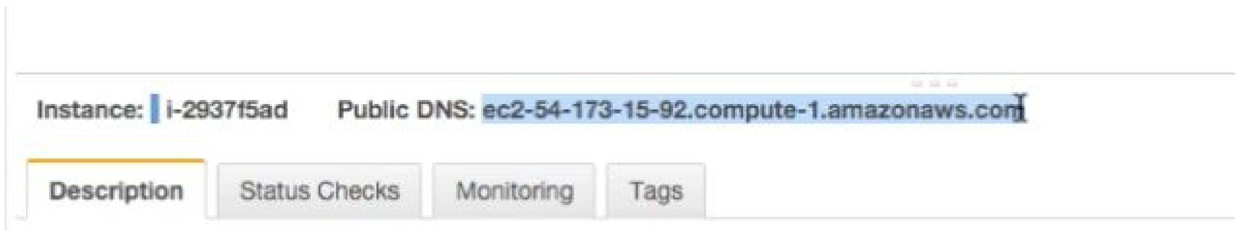
Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS
	i-2937f5ad	g2.2xlarge	us-east-1a	running	Initializing	None	ec2-54-173-1

Instance: i-2937f5ad　Public DNS: ec2-54-173-15-92.compute-1.amazonaws.com

Instance ID i-2937f5ad　Public DNS ec2-54-173-15-92.compute-

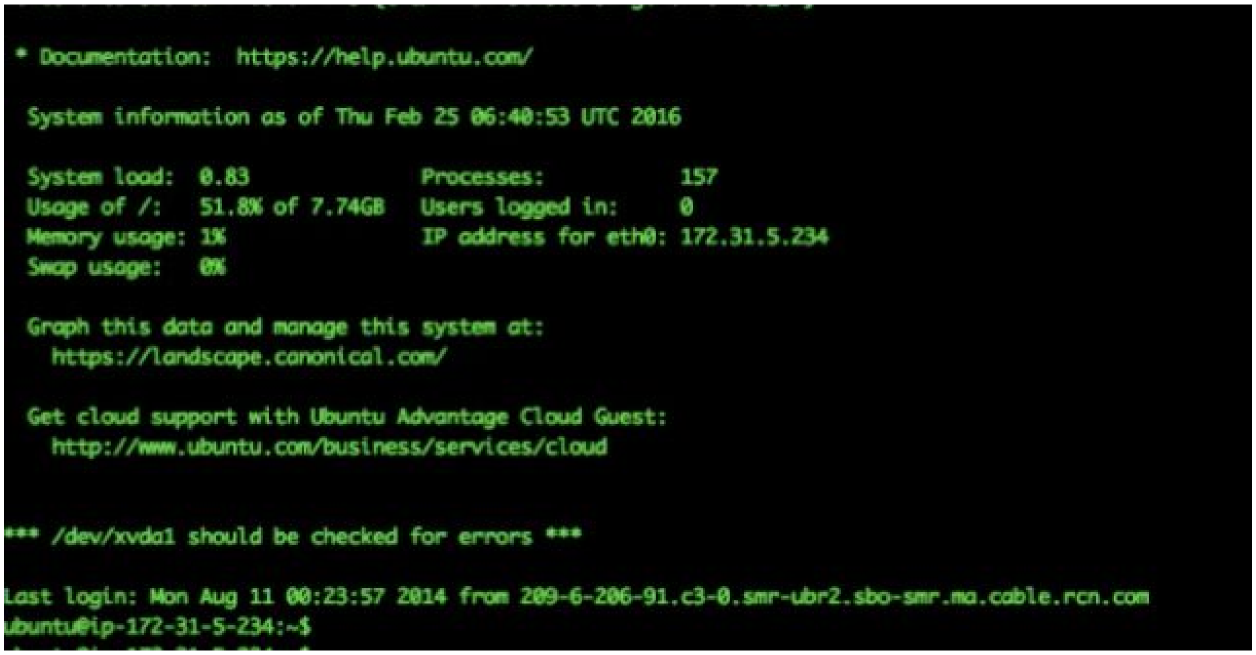
Address to the machine is provided as the **Public DNS** and log in to the machine where name of the **operating system** needs to be selected while creating the stop instances. Enter in the folder where the **`awstheano_name.pem`** located, cd in the folder and type in the terminal,

```
> chmod 400 awstheano_name.pem
> ssh -i awstheano_name.pem
ubuntu@ec2-54-173-15-92.compute-1.amazonaws.com
```

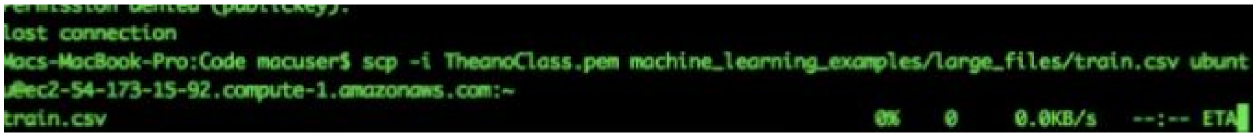


Instance: i-2937f5ad Public DNS: ec2-54-173-15-92.compute-1.amazonaws.com

After logged in the machine,



```
 * Documentation:  https://help.ubuntu.com/

  System information as of Thu Feb 25 06:40:53 UTC 2016

  System load:  0.83              Processes:           157
  Usage of /:   51.8% of 7.74GB   Users logged in:     0
  Memory usage: 1%                IP address for eth0: 172.31.5.234
  Swap usage:   0%

  Graph this data and manage this system at:
    https://landscape.canonical.com/

  Get cloud support with Ubuntu Advantage Cloud Guest:
    http://www.ubuntu.com/business/services/cloud


*** /dev/xvda1 should be checked for errors ***

Last login: Mon Aug 11 00:23:57 2014 from 209-6-206-91.c3-0.smr-ubr2.sbo-smr.ma.cable.rcn.com
ubuntu@ip-172-31-5-234:~$
```



```
lost connection
Macs-MacBook-Pro:Code macuser$ scp -i TheanoClass.pem machine_learning_examples/large_files/train.csv ubunt
u@ec2-54-173-15-92.compute-1.amazonaws.com:~
train.csv                                                    0%    0     0.0KB/s   --:-- ETA
```

Copy the train data to the **EC2 AWS**,

> scp -i awstheano_name.pem machine_learning_examples/large_files/train.csv ubuntu@ec2-54-173-15-92.compute-1.amazonaws.com

Install the **git** in the AWS cloud if necessary,
> sudo apt-get install git

Install the **pandas** in the AWS cloud if necessary,
> sudo easy_install pandas

Clone with the **SSH**
> git clone git@github.com:lazyprogrammer/machine_learning_examples.git

ISSUE: the GIT may take long time to install. So, it's better to copy over the file from the local sources. The following command will copy all the files from the **[ann_class2] folder to the AWS cloud.** After **cd** in the **machine_learning_examples** folder, run the command in the terminal,

> **scp -i awstheano_name.pem machine_learning_examples/ann_class2/* ubuntu@ec2-54-173-15-92.compute-1.amazonaws.com**

> ssh -i awstheano_name.pem ubuntu@ec2-54-173-15-92.compute-1.amazonaws.com

# check what's in the folder
> ls

# make a new folder in the AWS cloud,
> mkdir ann_class2

Copy all the files that comes from the local machine to the [ann_class2] folder in the AWS cloud.

# move all the **.py** files in the newly created **ann_class2** folder,
> mv *.py ann_class2/

> cd ann_class2

Need to figure out how to open the files in the Ubuntu operating system in the AWS. Make sure all the data types in the **theano_gpu.py** file are in 32 bytes,

```
    Ytrain_ind = y2indicator(Ytrain).astype(np.float32)
    Ytest_ind = y2indicator(Ytest).astype(np.float32)
```

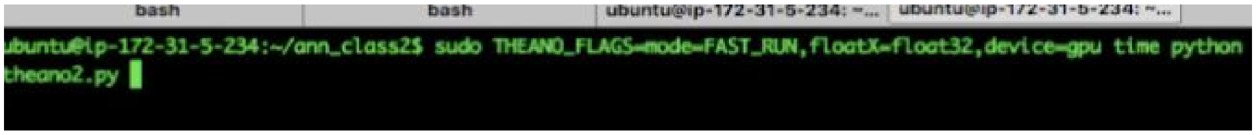
Then, quit with the command **Q**

> vi theano2.py

# run the command to execute the python file in the AWS cloud,
> sudo python theano2.py

# run the **theano2.py** file in the CPU AWS
> sudo THEANO_FLAGS=mode=FAST_RUN,device=cpu,floatX=float32 time python theano2.py

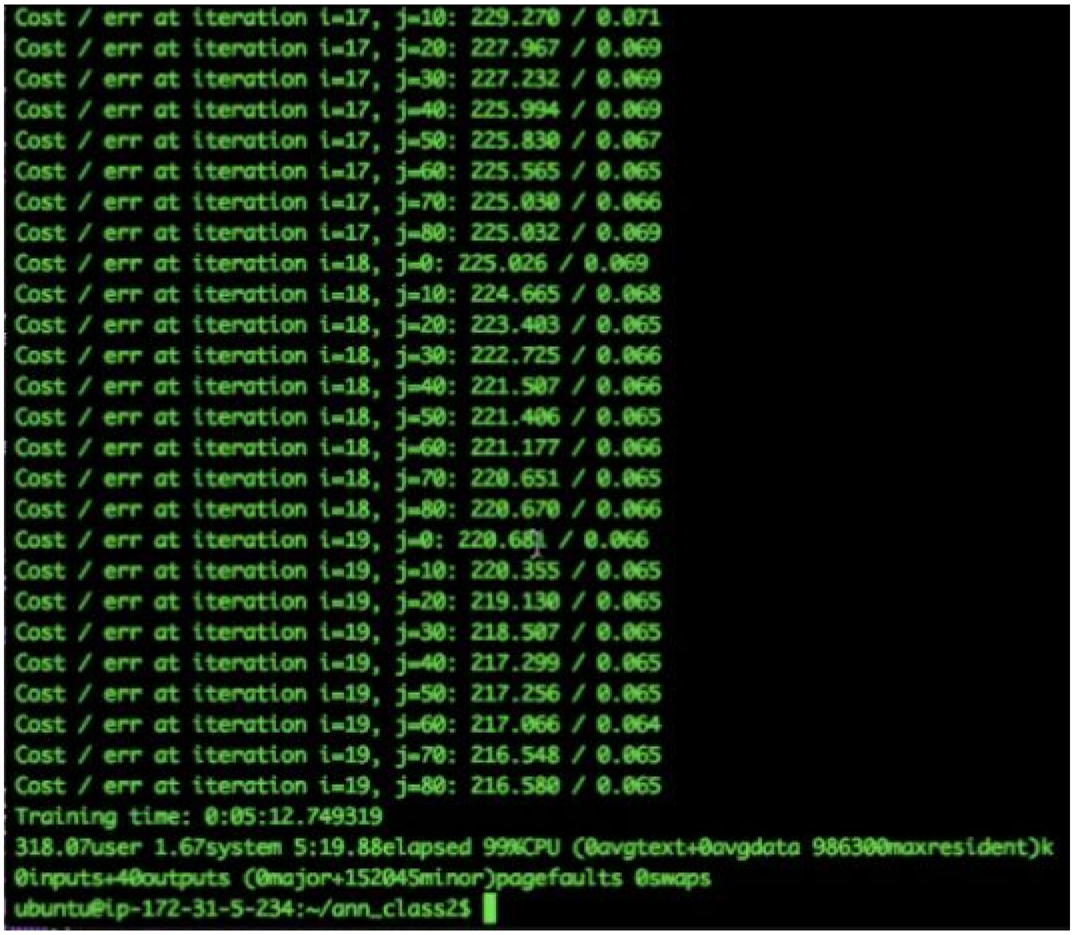
# Get rid of the matplotlib and run the **theano_gpu.py** file in the CPU AWS



```
ubuntu@ip-172-31-5-234:~/ann_class2$ sudo THEANO_FLAGS=mode=FAST_RUN,floatX=float32,device=gpu time python
theano2.py
```

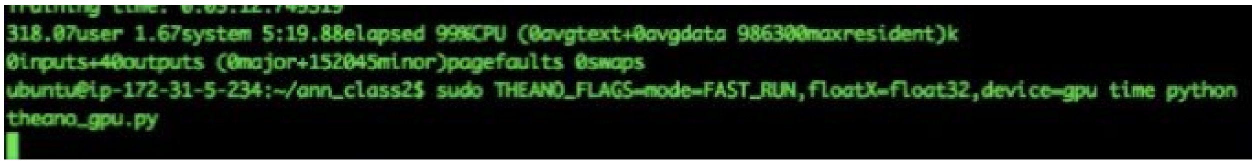
> sudo THEANO_FLAGS=mode=FAST_RUN,device=cpu,floatX=float32 time python theano_gpu.py

# with the CPU, the total time is about 5 minutes for the training time.



```
Cost / err at iteration i=17, j=10: 229.270 / 0.071
Cost / err at iteration i=17, j=20: 227.967 / 0.069
Cost / err at iteration i=17, j=30: 227.232 / 0.069
Cost / err at iteration i=17, j=40: 225.994 / 0.069
Cost / err at iteration i=17, j=50: 225.830 / 0.067
Cost / err at iteration i=17, j=60: 225.565 / 0.065
Cost / err at iteration i=17, j=70: 225.030 / 0.066
Cost / err at iteration i=17, j=80: 225.032 / 0.069
Cost / err at iteration i=18, j=0: 225.026 / 0.069
Cost / err at iteration i=18, j=10: 224.665 / 0.068
Cost / err at iteration i=18, j=20: 223.403 / 0.065
Cost / err at iteration i=18, j=30: 222.725 / 0.066
Cost / err at iteration i=18, j=40: 221.507 / 0.066
Cost / err at iteration i=18, j=50: 221.406 / 0.065
Cost / err at iteration i=18, j=60: 221.177 / 0.066
Cost / err at iteration i=18, j=70: 220.651 / 0.065
Cost / err at iteration i=18, j=80: 220.670 / 0.066
Cost / err at iteration i=19, j=0: 220.681 / 0.066
Cost / err at iteration i=19, j=10: 220.355 / 0.065
Cost / err at iteration i=19, j=20: 219.130 / 0.065
Cost / err at iteration i=19, j=30: 218.507 / 0.065
Cost / err at iteration i=19, j=40: 217.299 / 0.065
Cost / err at iteration i=19, j=50: 217.256 / 0.065
Cost / err at iteration i=19, j=60: 217.066 / 0.064
Cost / err at iteration i=19, j=70: 216.548 / 0.065
Cost / err at iteration i=19, j=80: 216.580 / 0.065
Training time: 0:05:12.749319
318.07user 1.67system 5:19.88elapsed 99%CPU (0avgtext+0avgdata 986300maxresident)k
0inputs+40outputs (0major+152045minor)pagefaults 0swaps
ubuntu@ip-172-31-5-234:~/ann_class2$
```

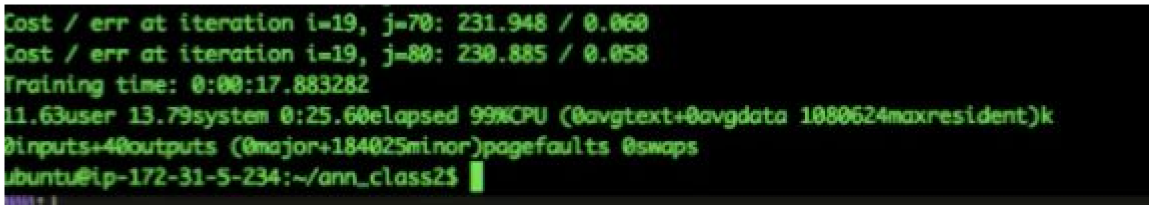
# now, run the **theano_gpu.py** file in the GPU AWS



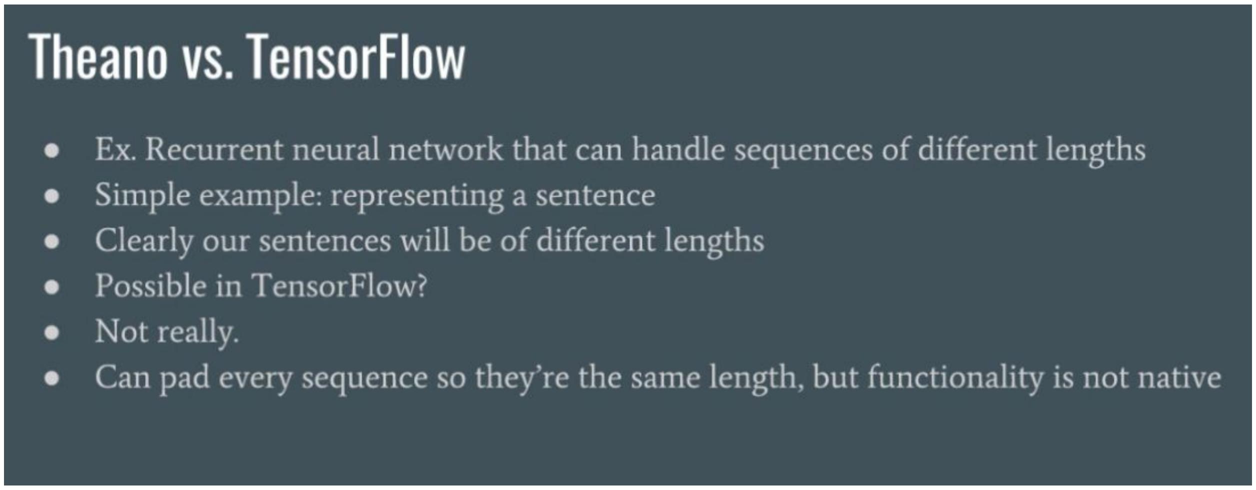
```
Training time: 0:05:12.749519
318.07user 1.67system 5:19.88elapsed 99%CPU (0avgtext+0avgdata 986300maxresident)k
0inputs+40outputs (0major+152045minor)pagefaults 0swaps
ubuntu@ip-172-31-5-234:~/ann_class2$ sudo THEANO_FLAGS=mode=FAST_RUN,floatX=float32,device=gpu time python
theano_gpu.py
```

> sudo THEANO_FLAGS=mode=FAST_RUN,device=gpu,floatX=float32 time python theano_gpu.py

# with the GPU, the total training time is about 17 seconds which is much better than the CPU.

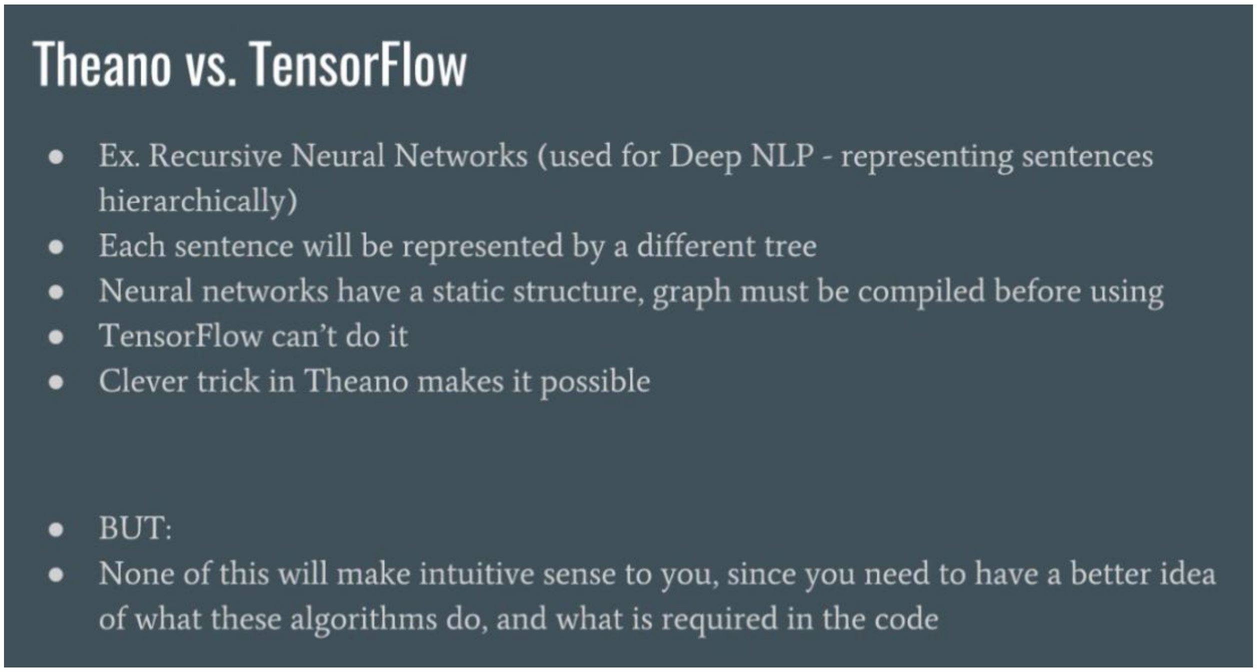


```
Cost / err at iteration i=19, j=70: 231.948 / 0.060
Cost / err at iteration i=19, j=80: 230.885 / 0.058
Training time: 0:00:17.883282
11.63user 13.79system 0:25.60elapsed 99%CPU (0avgtext+0avgdata 1080624maxresident)k
0inputs+40outputs (0major+184025minor)pagefaults 0swaps
ubuntu@ip-172-31-5-234:~/ann_class2$
```



Theano vs. TensorFlow

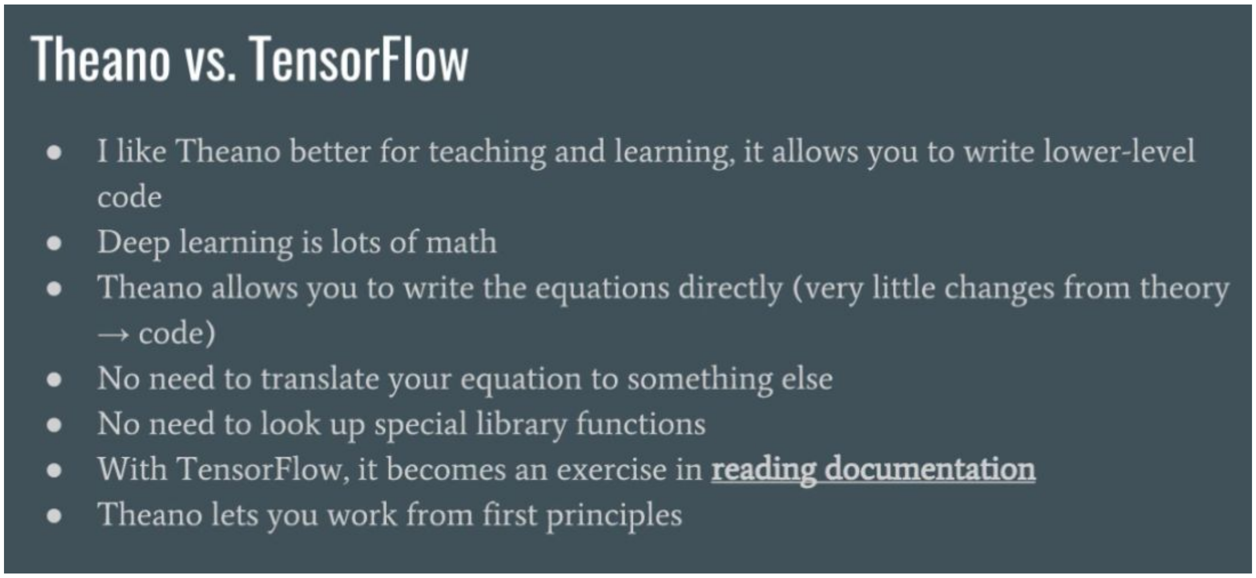
- Ex. Recurrent neural network that can handle sequences of different lengths
- Simple example: representing a sentence
- Clearly our sentences will be of different lengths
- Possible in TensorFlow?
- Not really.
- Can pad every sequence so they're the same length, but functionality is not native



Theano vs. TensorFlow

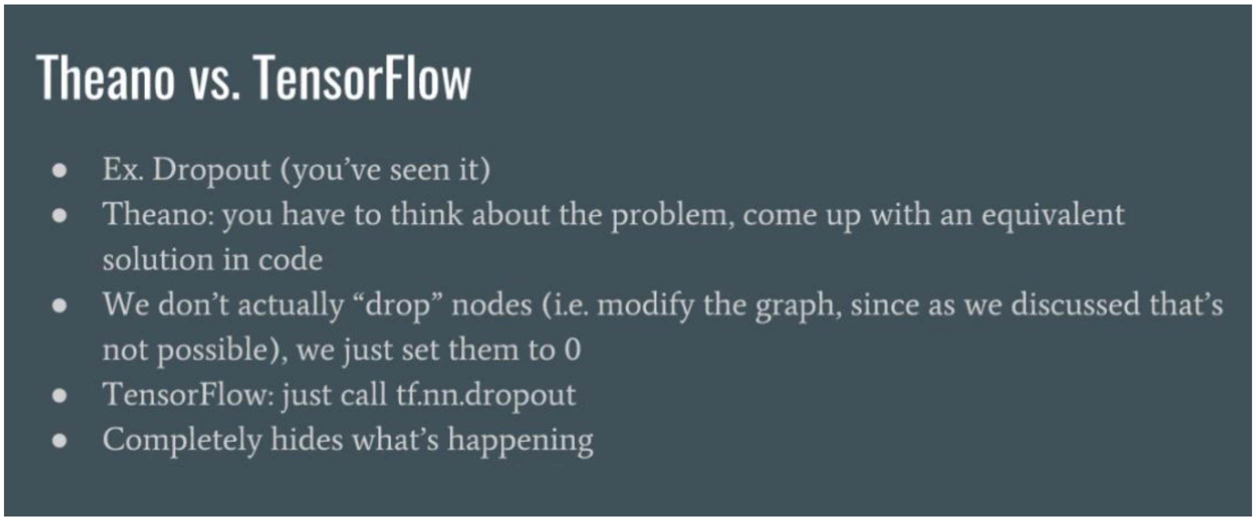
- Ex. Recursive Neural Networks (used for Deep NLP - representing sentences hierarchically)
- Each sentence will be represented by a different tree
- Neural networks have a static structure, graph must be compiled before using
- TensorFlow can't do it
- Clever trick in Theano makes it possible

- BUT:
- None of this will make intuitive sense to you, since you need to have a better idea of what these algorithms do, and what is required in the code



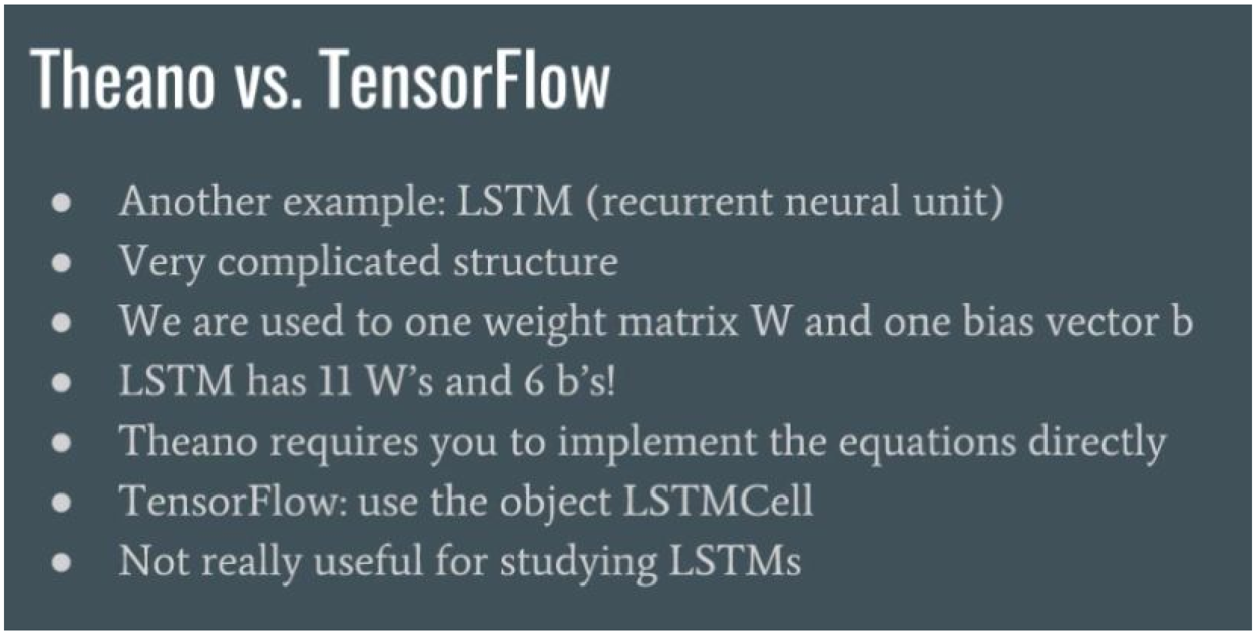
Theano vs. TensorFlow

- I like Theano better for teaching and learning, it allows you to write lower-level code
- Deep learning is lots of math
- Theano allows you to write the equations directly (very little changes from theory → code)
- No need to translate your equation to something else
- No need to look up special library functions
- With TensorFlow, it becomes an exercise in **reading documentation**
- Theano lets you work from first principles



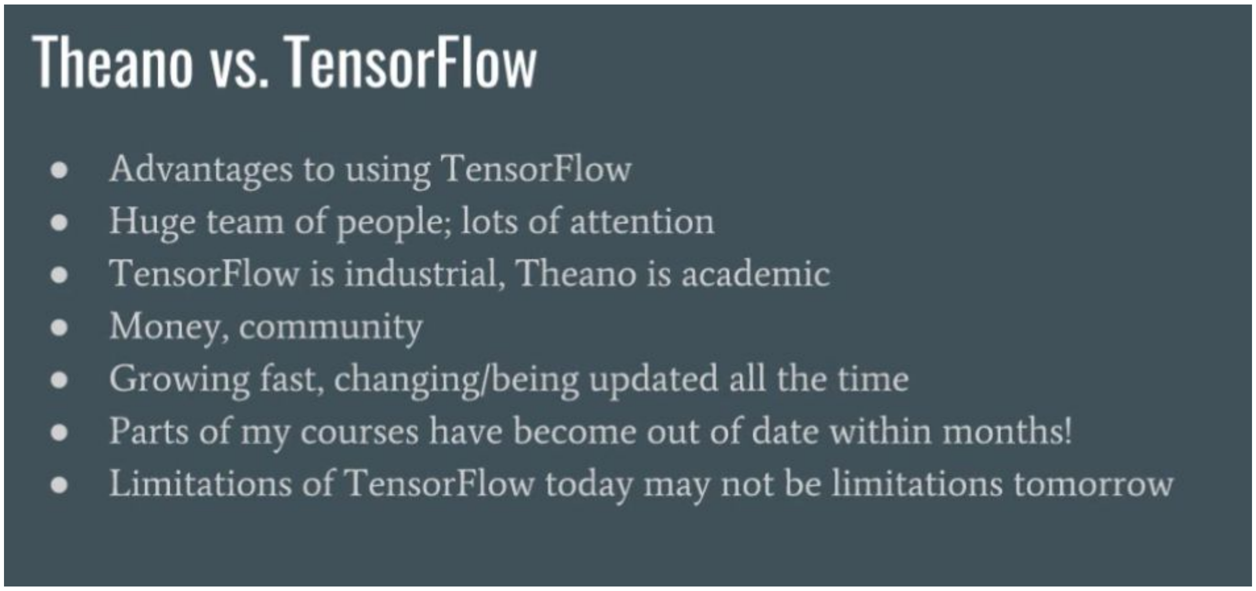
Theano vs. TensorFlow

- Ex. Dropout (you've seen it)
- Theano: you have to think about the problem, come up with an equivalent solution in code
- We don't actually "drop" nodes (i.e. modify the graph, since as we discussed that's not possible), we just set them to 0
- TensorFlow: just call tf.nn.dropout
- Completely hides what's happening



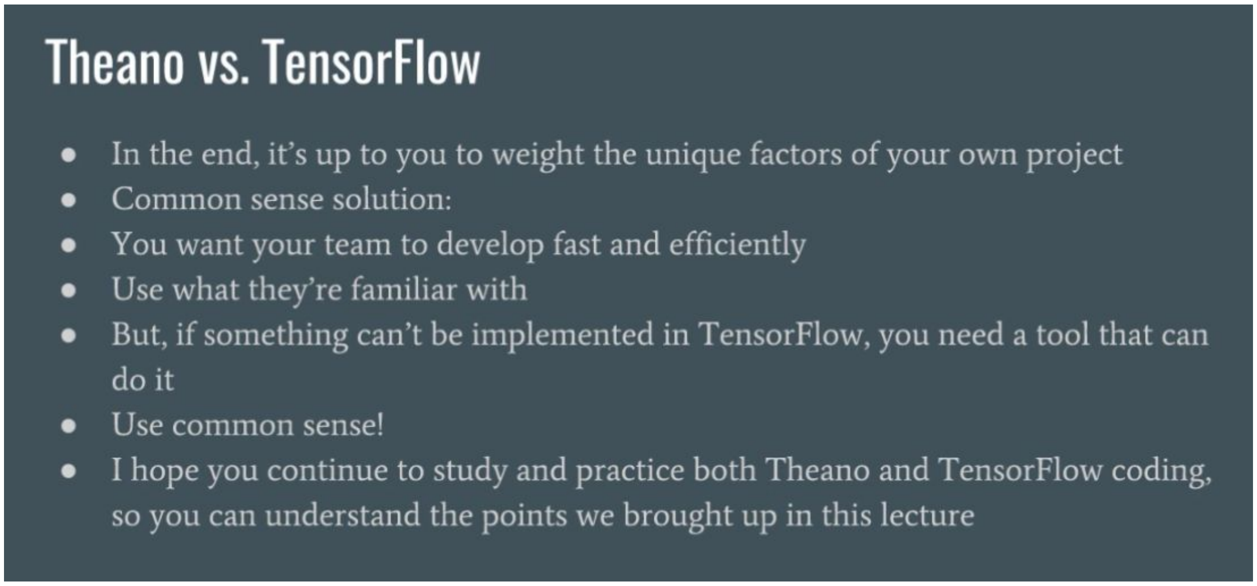
Theano vs. TensorFlow

- Another example: LSTM (recurrent neural unit)
- Very complicated structure
- We are used to one weight matrix W and one bias vector b
- LSTM has 11 W's and 6 b's!
- Theano requires you to implement the equations directly
- TensorFlow: use the object LSTMCell
- Not really useful for studying LSTMs



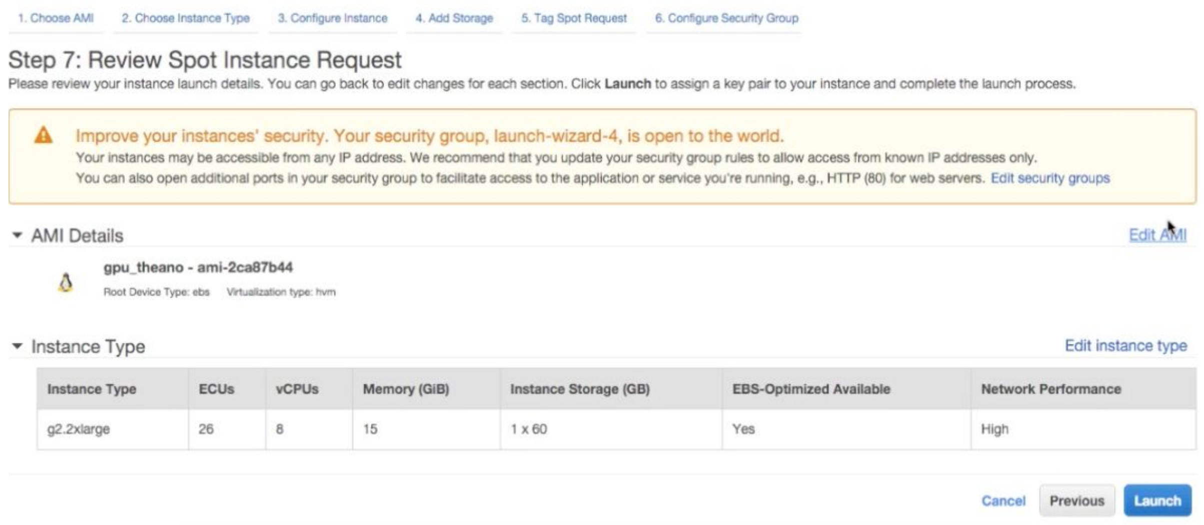
Theano vs. TensorFlow

- Advantages to using TensorFlow
- Huge team of people; lots of attention
- TensorFlow is industrial, Theano is academic
- Money, community
- Growing fast, changing/being updated all the time
- Parts of my courses have become out of date within months!
- Limitations of TensorFlow today may not be limitations tomorrow



Theano vs. TensorFlow

- In the end, it's up to you to weight the unique factors of your own project
- Common sense solution:
- You want your team to develop fast and efficiently
- Use what they're familiar with
- But, if something can't be implemented in TensorFlow, you need a tool that can do it
- Use common sense!
- I hope you continue to study and practice both Theano and TensorFlow coding, so you can understand the points we brought up in this lecture


1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Tag Spot Request 6. Configure Security Group

Step 7: Review Spot Instance Request

Please review your instance launch details. You can go back to edit changes for each section. Click **Launch** to assign a key pair to your instance and complete the launch process.

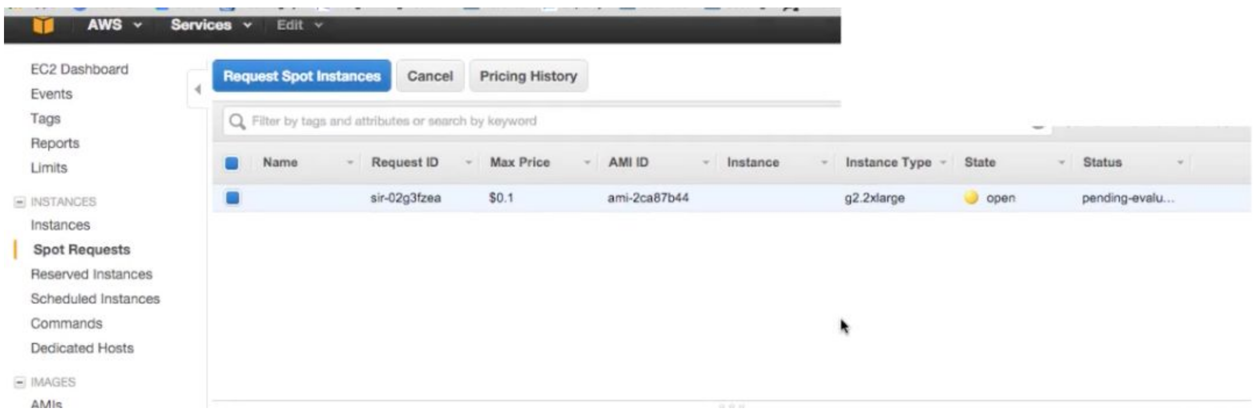
Improve your instances' security. Your security group, launch-wizard-4, is open to the world.
Your instances may be accessible from any IP address. We recommend that you update your security group rules to allow access from known IP addresses only.
You can also open additional ports in your security group to facilitate access to the application or service you're running, e.g., HTTP (80) for web servers. Edit security groups

AMI Details — Edit AMI

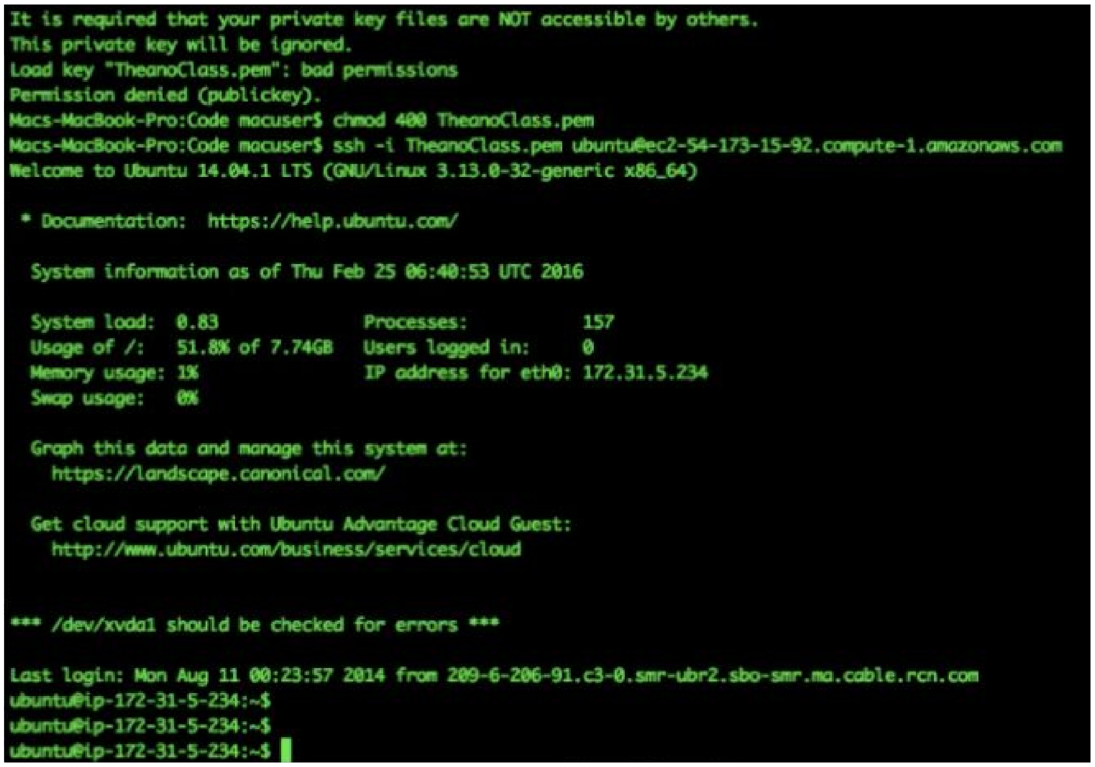
gpu_theano - ami-2ca87b44
Root Device Type: ebs Virtualization type: hvm

Instance Type — Edit instance type

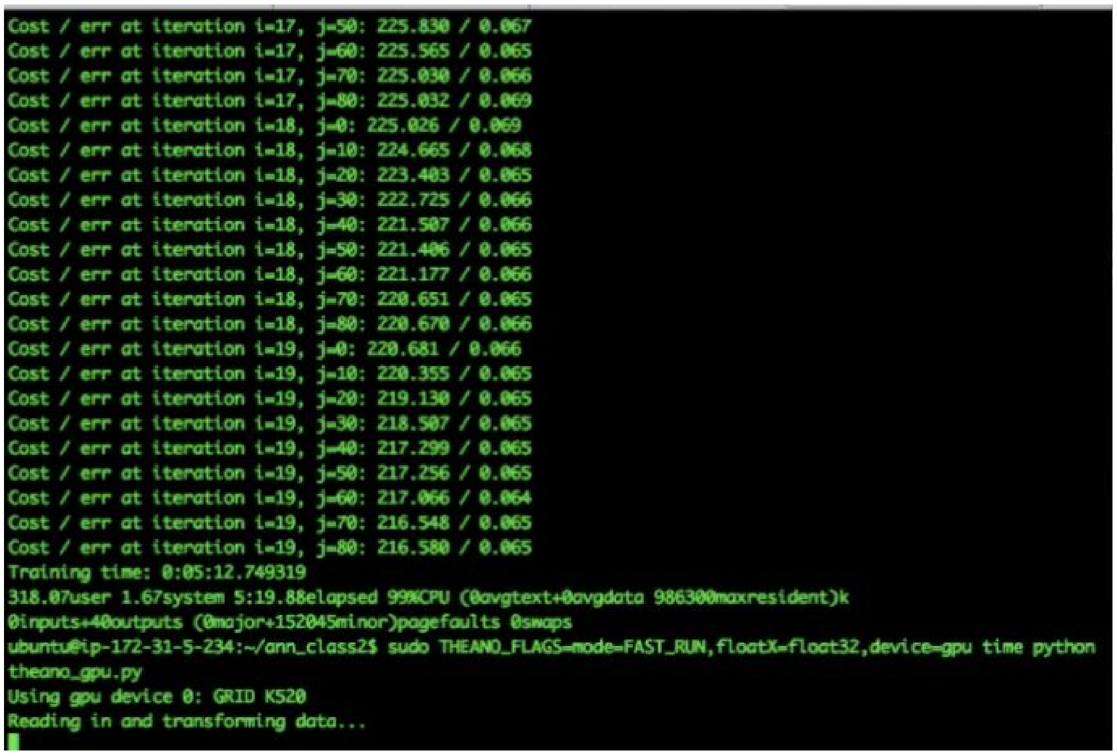
Instance Type	ECUs	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance
g2.2xlarge	26	8	15	1 x 60	Yes	High



Name	Request ID	Max Price	AMI ID	Instance	Instance Type	State	Status
	sir-02g3fzea	$0.1	ami-2ca87b44		g2.2xlarge	open	pending-evalu...



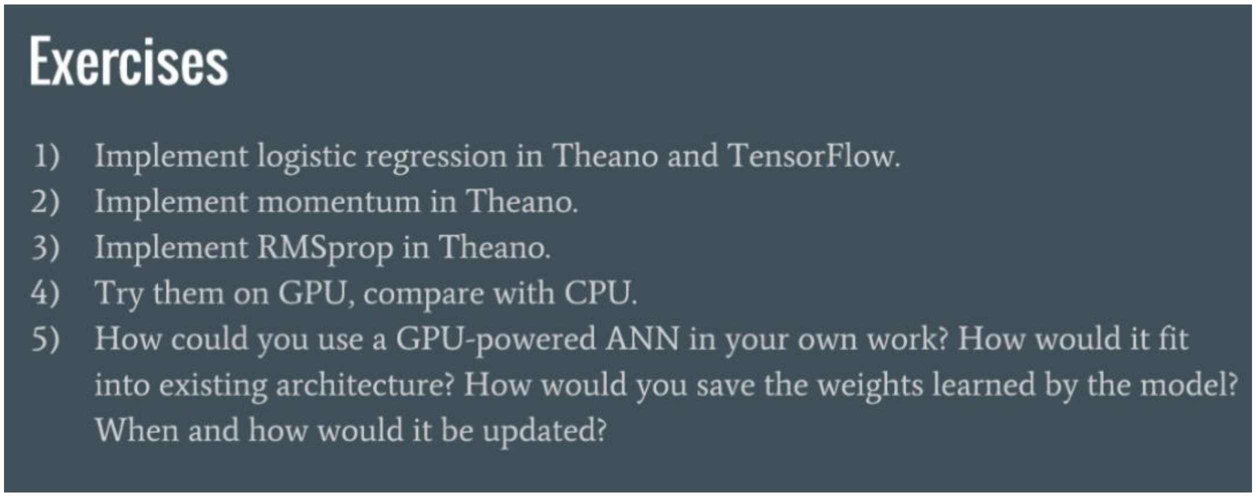
```
It is required that your private key files are NOT accessible by others.
This private key will be ignored.
Load key "TheanoClass.pem": bad permissions
Permission denied (publickey).
Macs-MacBook-Pro:Code macuser$ chmod 400 TheanoClass.pem
Macs-MacBook-Pro:Code macuser$ ssh -i TheanoClass.pem ubuntu@ec2-54-173-15-92.compute-1.amazonaws.com
Welcome to Ubuntu 14.04.1 LTS (GNU/Linux 3.13.0-32-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

  System information as of Thu Feb 25 06:40:53 UTC 2016

  System load:  0.83              Processes:           157
  Usage of /:   51.8% of 7.74GB   Users logged in:     0
  Memory usage: 1%                IP address for eth0: 172.31.5.234
  Swap usage:   0%

  Graph this data and manage this system at:
    https://landscape.canonical.com/

  Get cloud support with Ubuntu Advantage Cloud Guest:
    http://www.ubuntu.com/business/services/cloud


*** /dev/xvda1 should be checked for errors ***

Last login: Mon Aug 11 00:23:57 2014 from 209-6-206-91.c3-0.smr-ubr2.sbo-smr.ma.cable.rcn.com
ubuntu@ip-172-31-5-234:~$
ubuntu@ip-172-31-5-234:~$
ubuntu@ip-172-31-5-234:~$
```



```
Cost / err at iteration i=17, j=50: 225.830 / 0.067
Cost / err at iteration i=17, j=60: 225.565 / 0.065
Cost / err at iteration i=17, j=70: 225.030 / 0.066
Cost / err at iteration i=17, j=80: 225.032 / 0.069
Cost / err at iteration i=18, j=0: 225.026 / 0.069
Cost / err at iteration i=18, j=10: 224.665 / 0.068
Cost / err at iteration i=18, j=20: 223.403 / 0.065
Cost / err at iteration i=18, j=30: 222.725 / 0.066
Cost / err at iteration i=18, j=40: 221.507 / 0.066
Cost / err at iteration i=18, j=50: 221.406 / 0.065
Cost / err at iteration i=18, j=60: 221.177 / 0.066
Cost / err at iteration i=18, j=70: 220.651 / 0.065
Cost / err at iteration i=18, j=80: 220.670 / 0.066
Cost / err at iteration i=19, j=0: 220.681 / 0.066
Cost / err at iteration i=19, j=10: 220.355 / 0.065
Cost / err at iteration i=19, j=20: 219.130 / 0.065
Cost / err at iteration i=19, j=30: 218.507 / 0.065
Cost / err at iteration i=19, j=40: 217.299 / 0.065
Cost / err at iteration i=19, j=50: 217.256 / 0.065
Cost / err at iteration i=19, j=60: 217.066 / 0.064
Cost / err at iteration i=19, j=70: 216.548 / 0.065
Cost / err at iteration i=19, j=80: 216.580 / 0.065
Training time: 0:05:12.749319
318.07user 1.67system 5:19.88elapsed 99%CPU (0avgtext+0avgdata 986300maxresident)k
0inputs+40outputs (0major+152045minor)pagefaults 0swaps
ubuntu@ip-172-31-5-234:~/ann_class2$ sudo THEANO_FLAGS=mode=FAST_RUN,floatX=float32,device=gpu time python
theano_gpu.py
Using gpu device 0: GRID K520
Reading in and transforming data...
```

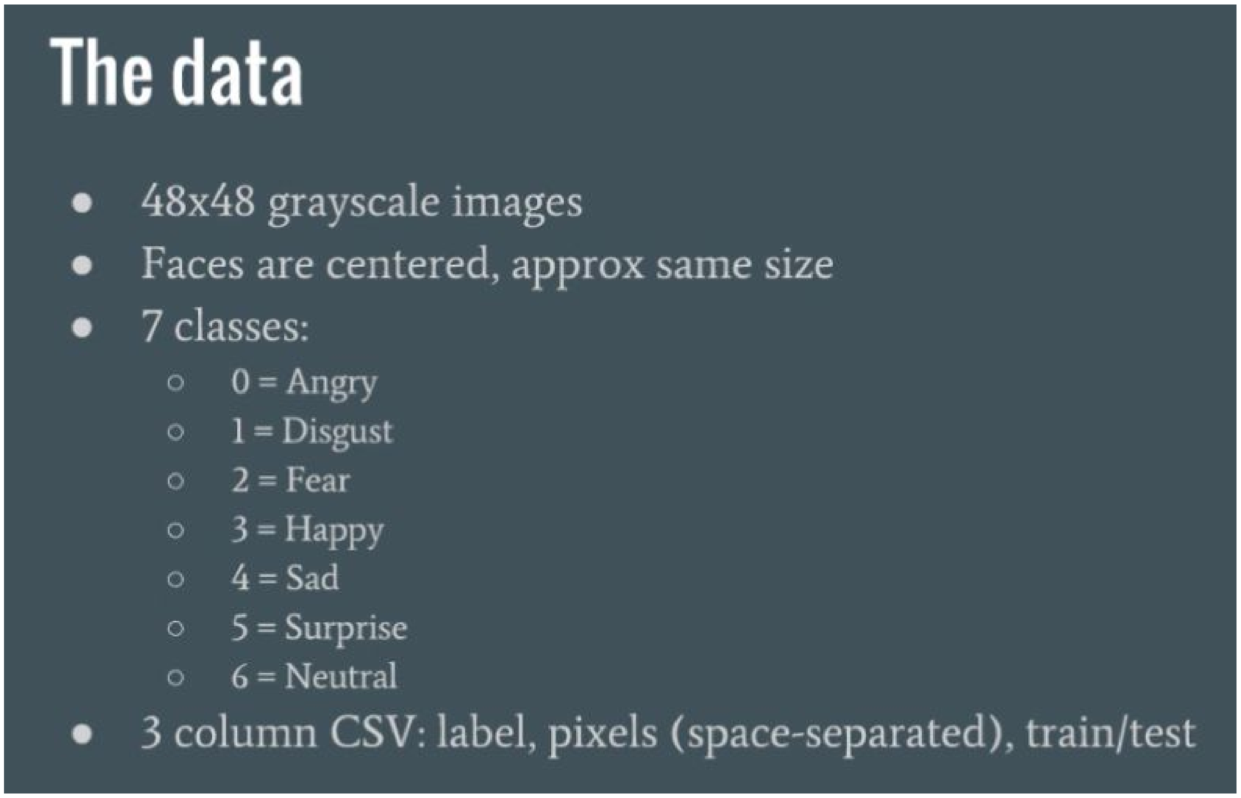
CPU

Set up a GPU instance in the Amazon Web Service (AWS)



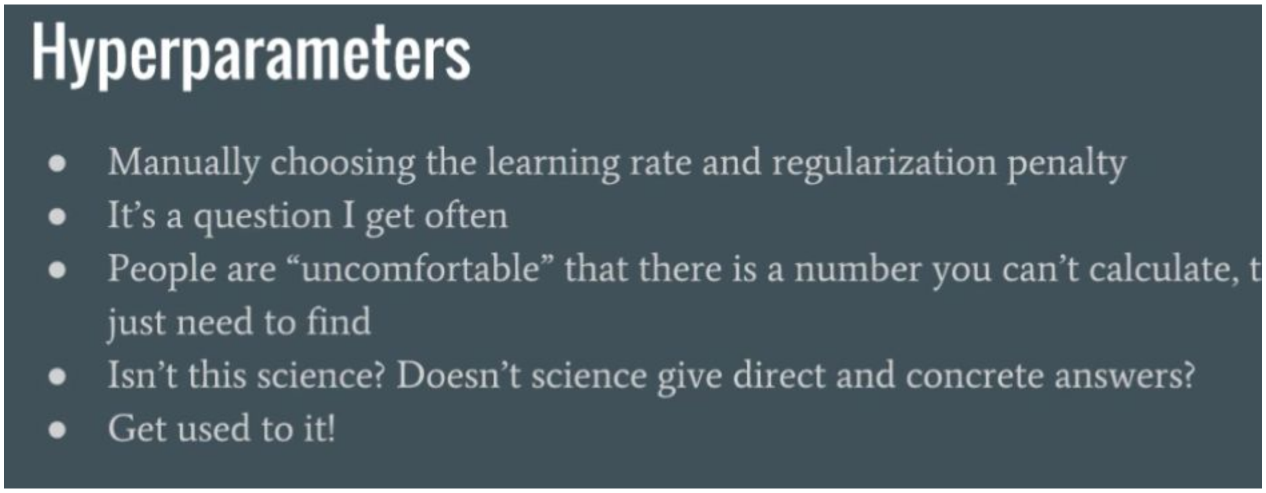
Exercises

1) Implement logistic regression in Theano and TensorFlow.
2) Implement momentum in Theano.
3) Implement RMSprop in Theano.
4) Try them on GPU, compare with CPU.
5) How could you use a GPU-powered ANN in your own work? How would it fit into existing architecture? How would you save the weights learned by the model? When and how would it be updated?



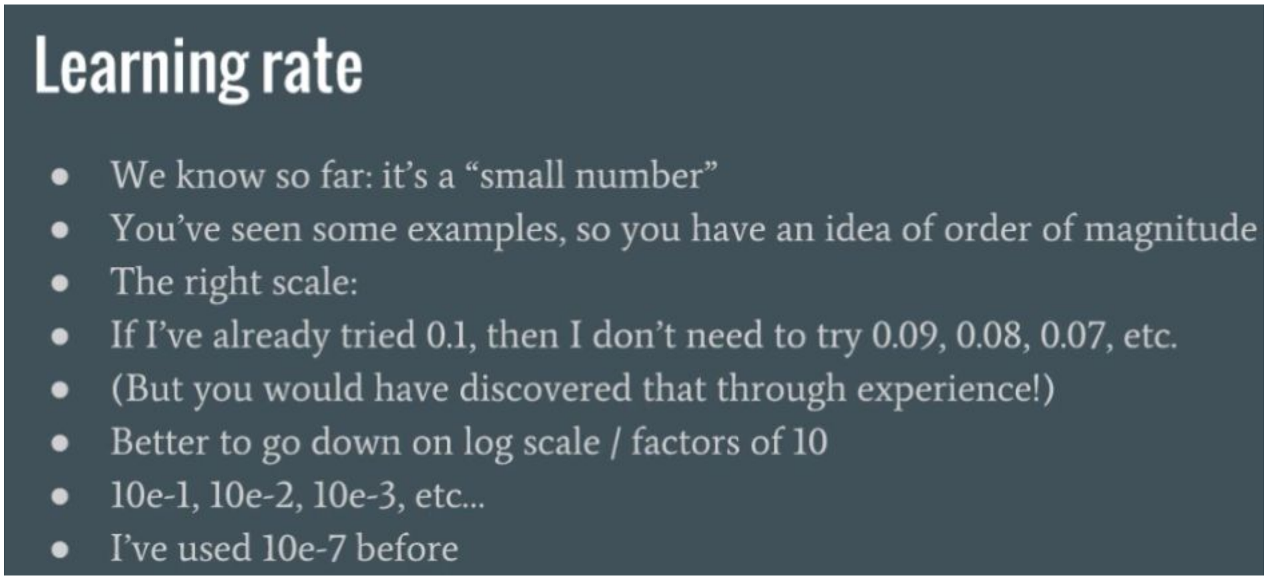
The data

- 48x48 grayscale images
- Faces are centered, approx same size
- 7 classes:
  - 0 = Angry
  - 1 = Disgust
  - 2 = Fear
  - 3 = Happy
  - 4 = Sad
  - 5 = Surprise
  - 6 = Neutral
- 3 column CSV: label, pixels (space-separated), train/test



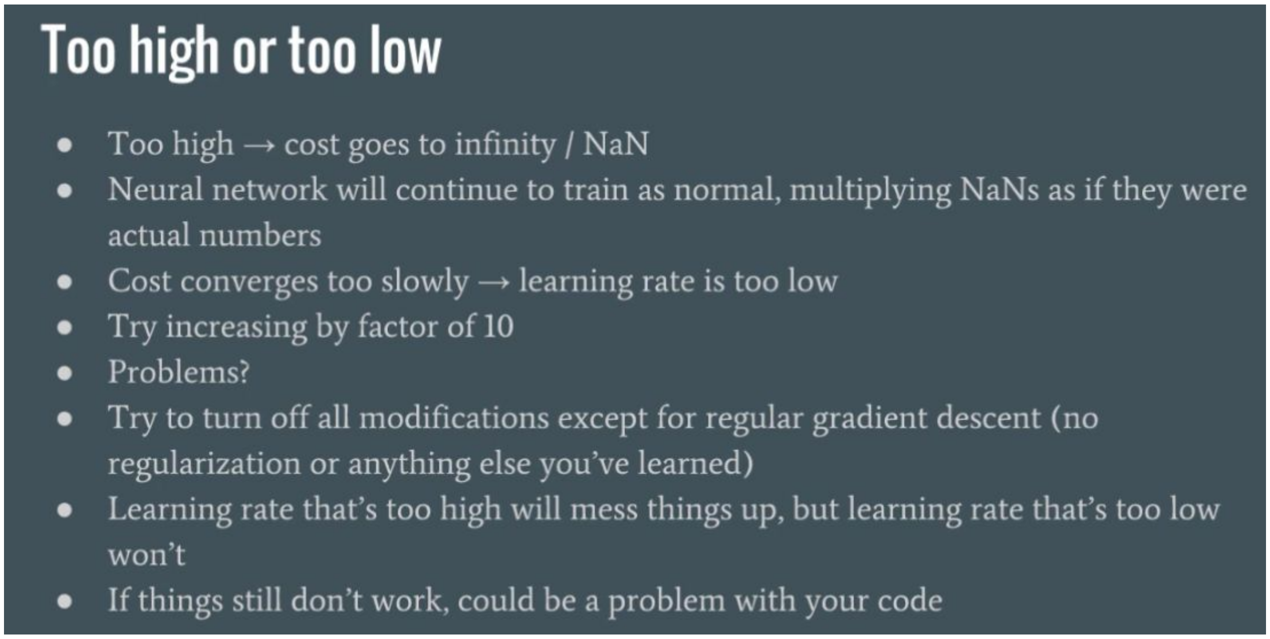
Hyperparameters

- Manually choosing the learning rate and regularization penalty
- It's a question I get often
- People are "uncomfortable" that there is a number you can't calculate, t just need to find
- Isn't this science? Doesn't science give direct and concrete answers?
- Get used to it!



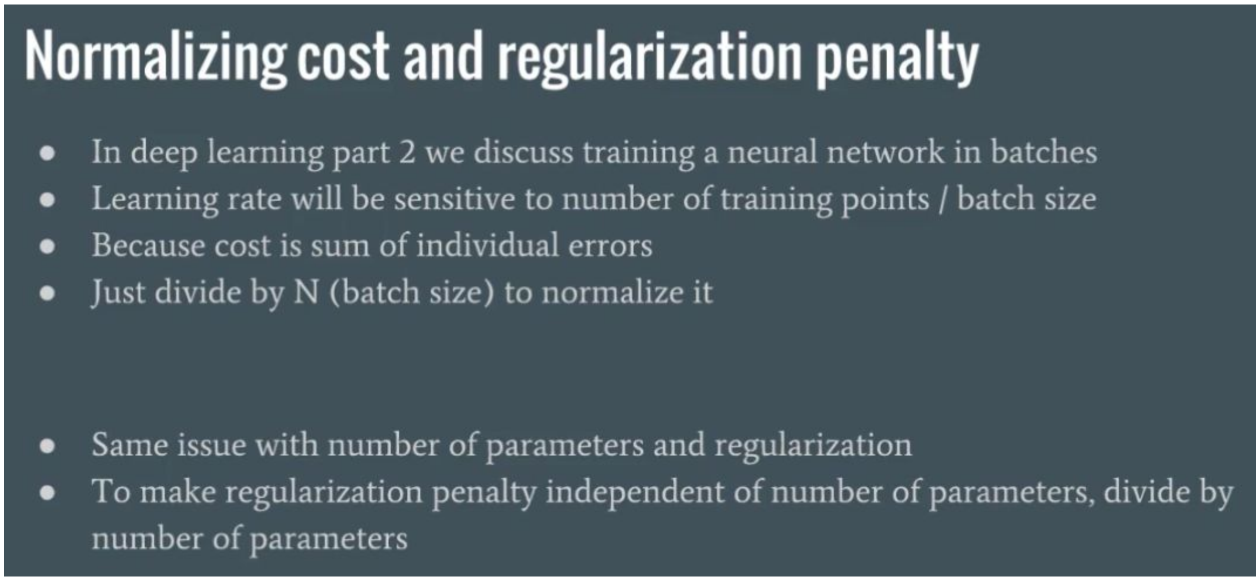
Learning rate

- We know so far: it's a "small number"
- You've seen some examples, so you have an idea of order of magnitude
- The right scale:
- If I've already tried 0.1, then I don't need to try 0.09, 0.08, 0.07, etc.
- (But you would have discovered that through experience!)
- Better to go down on log scale / factors of 10
- 10e-1, 10e-2, 10e-3, etc...
- I've used 10e-7 before



Too high or too low

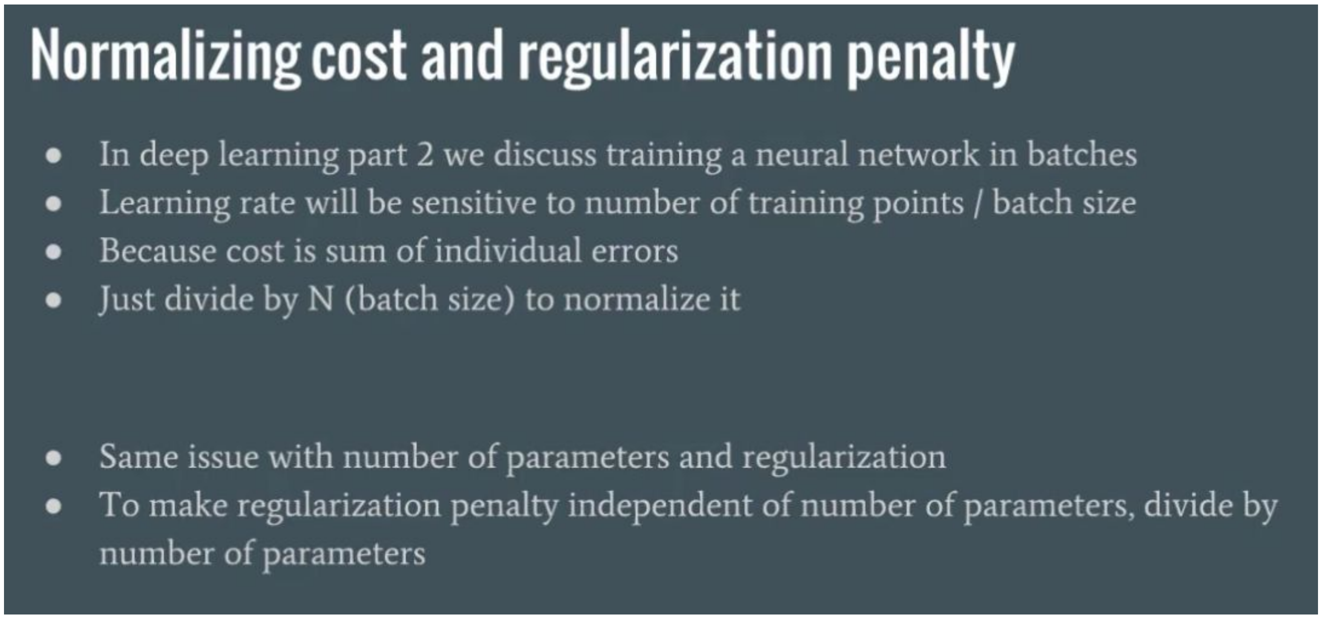
- Too high → cost goes to infinity / NaN
- Neural network will continue to train as normal, multiplying NaNs as if they were actual numbers
- Cost converges too slowly → learning rate is too low
- Try increasing by factor of 10
- Problems?
- Try to turn off all modifications except for regular gradient descent (no regularization or anything else you've learned)
- Learning rate that's too high will mess things up, but learning rate that's too low won't
- If things still don't work, could be a problem with your code



Normalizing cost and regularization penalty

- In deep learning part 2 we discuss training a neural network in batches
- Learning rate will be sensitive to number of training points / batch size
- Because cost is sum of individual errors
- Just divide by N (batch size) to normalize it

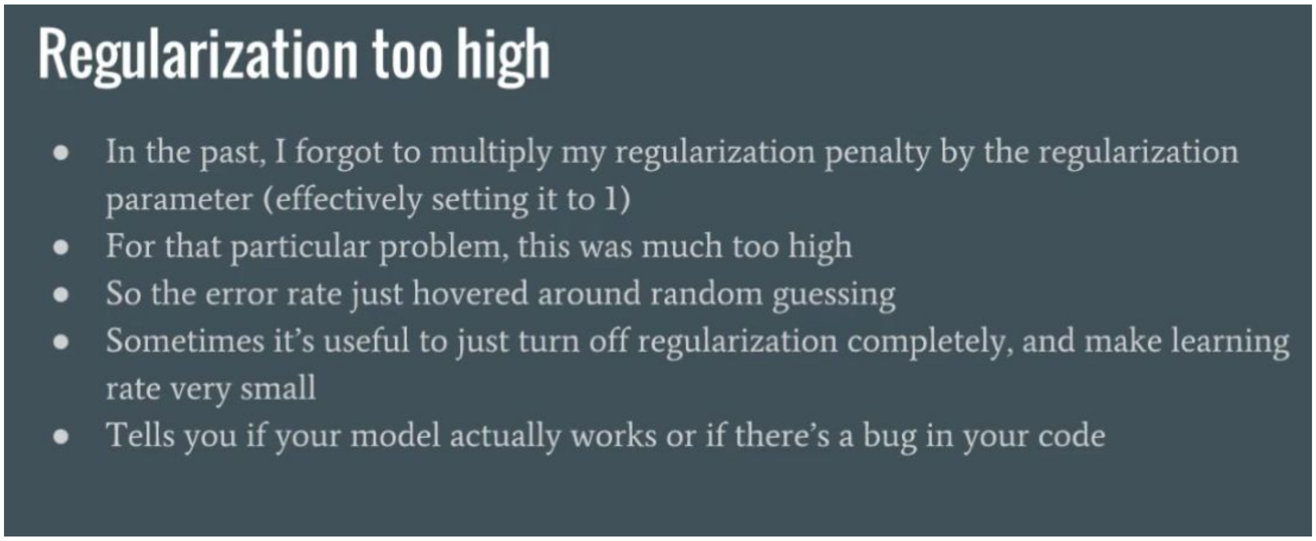
- Same issue with number of parameters and regularization
- To make regularization penalty independent of number of parameters, divide by number of parameters



Normalizing cost and regularization penalty

- In deep learning part 2 we discuss training a neural network in batches
- Learning rate will be sensitive to number of training points / batch size
- Because cost is sum of individual errors
- Just divide by N (batch size) to normalize it

- Same issue with number of parameters and regularization
- To make regularization penalty independent of number of parameters, divide by number of parameters



Regularization too high

- In the past, I forgot to multiply my regularization penalty by the regularization parameter (effectively setting it to 1)
- For that particular problem, this was much too high
- So the error rate just hovered around random guessing
- Sometimes it's useful to just turn off regularization completely, and make learning rate very small
- Tells you if your model actually works or if there's a bug in your code