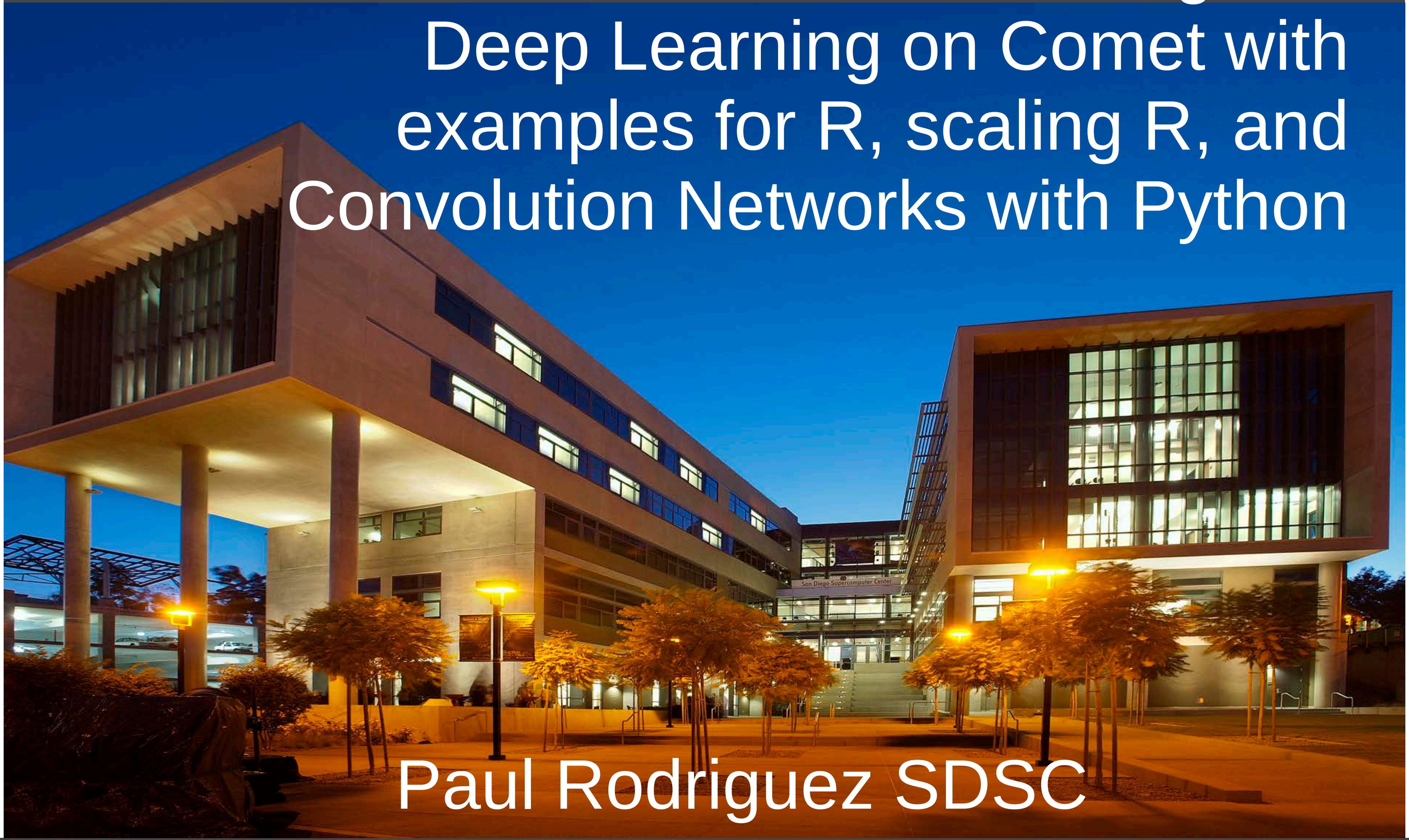
Introductions to Machine Learning and Deep Learning on Comet with examples for R, scaling R, and Convolution Networks with Python

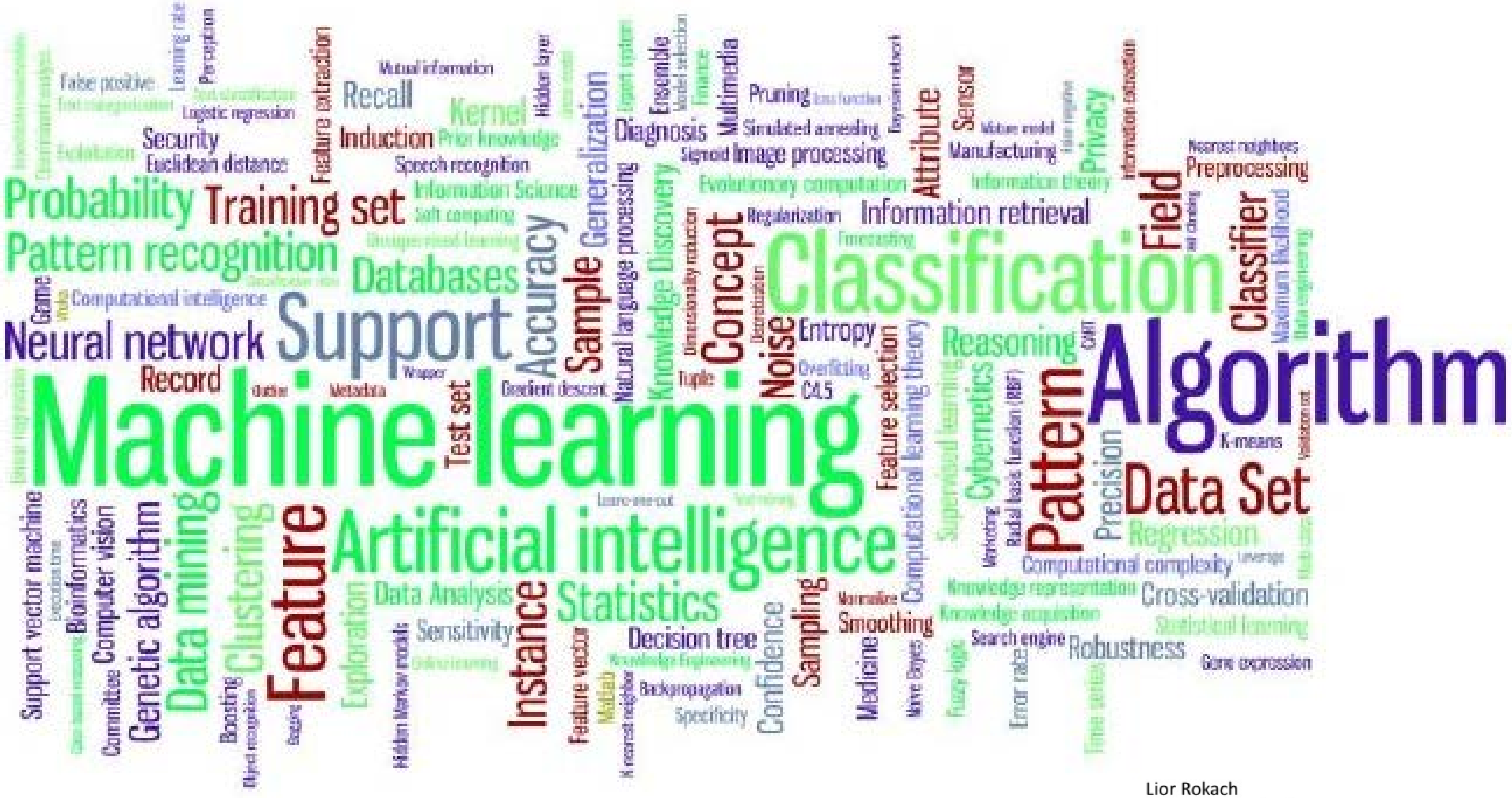


Paul Rodriguez SDSC

Outline

I. Analytics, Machine Learning in a Nutshell
II. R, Scaling in R, Parallel R
III. Deep Learning in a Nutshell
IV. Deep Learning Tutorial

Lots of Terms:



Lior Rokach
Department of Information Systems Engineering
Ben-Gurion University of the Negev

Lots of Terms: what are the key ideas?

improving system performance with data

e.g. statistical learning,

e.g. models with algorithms for fitting parameters

Machine Learning Models

- Classification
- Regression/Predictive

Supervised (dependent variable or outcome labels given)

- Cluster
- Matrix Factorization

Unsupervised (no labels)

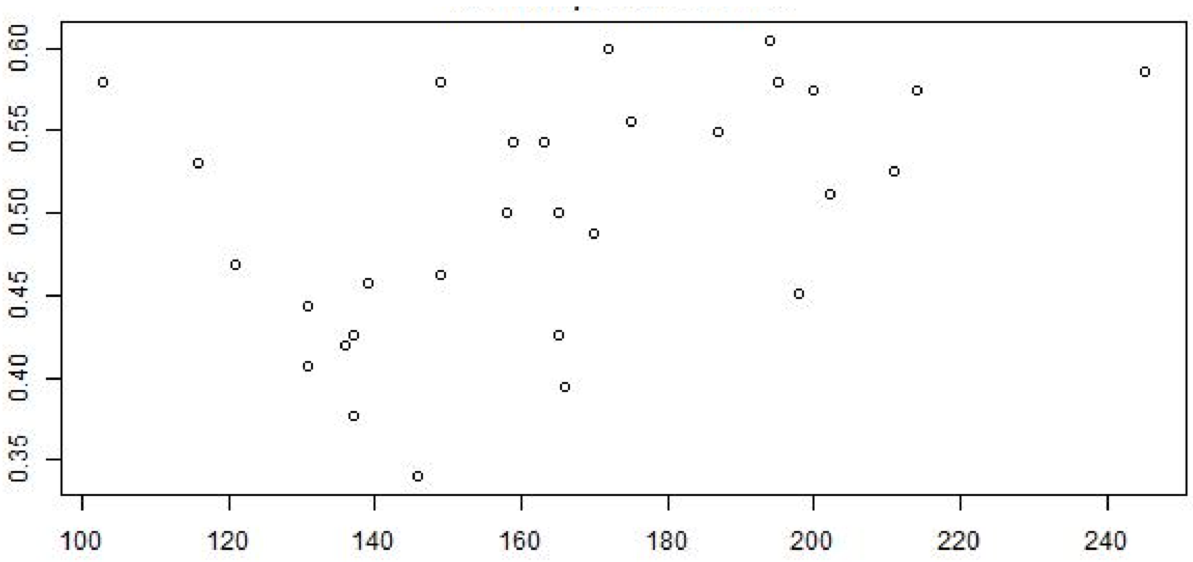
- Bayesian (i.e. learning probability distributions)

Statistical but comes up in HPC settings

A data example:Home Runs and W-L percent

W-L %

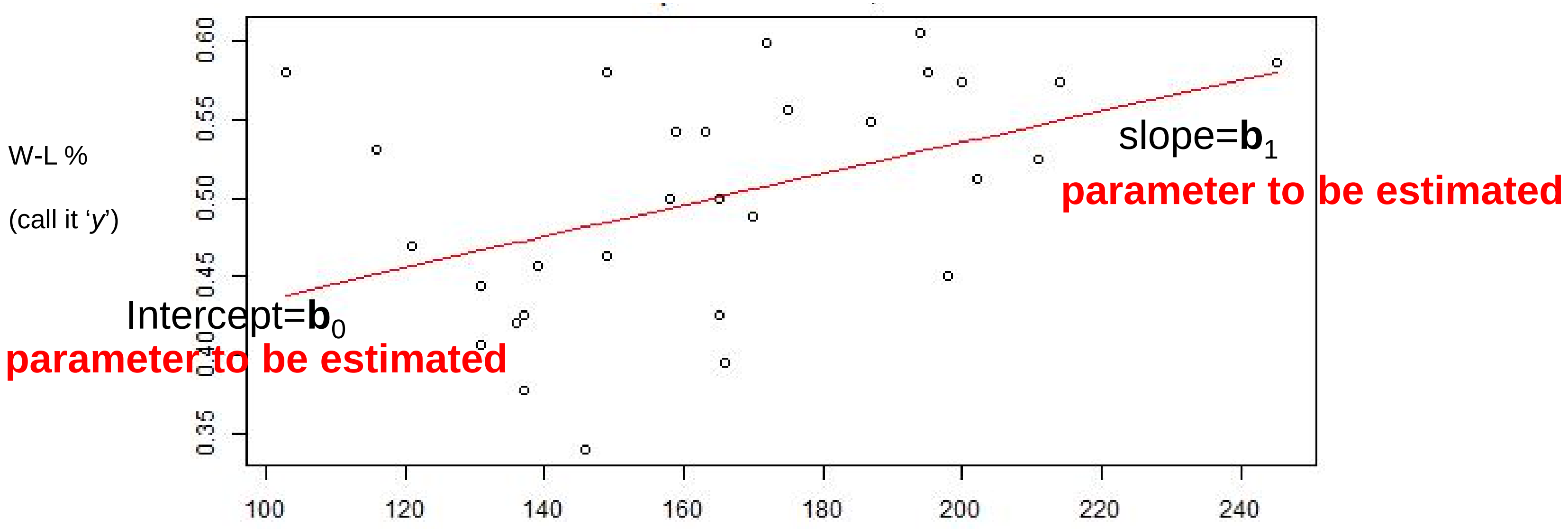
(call it ‘*y*’)



Number of Team HRs 2012 (call it ‘*x*’)

Recall Linear Regression is Fitting a Line – to minimize error

the Model: $y = f(x, b) = b_0 * 1 + b_1 * x$

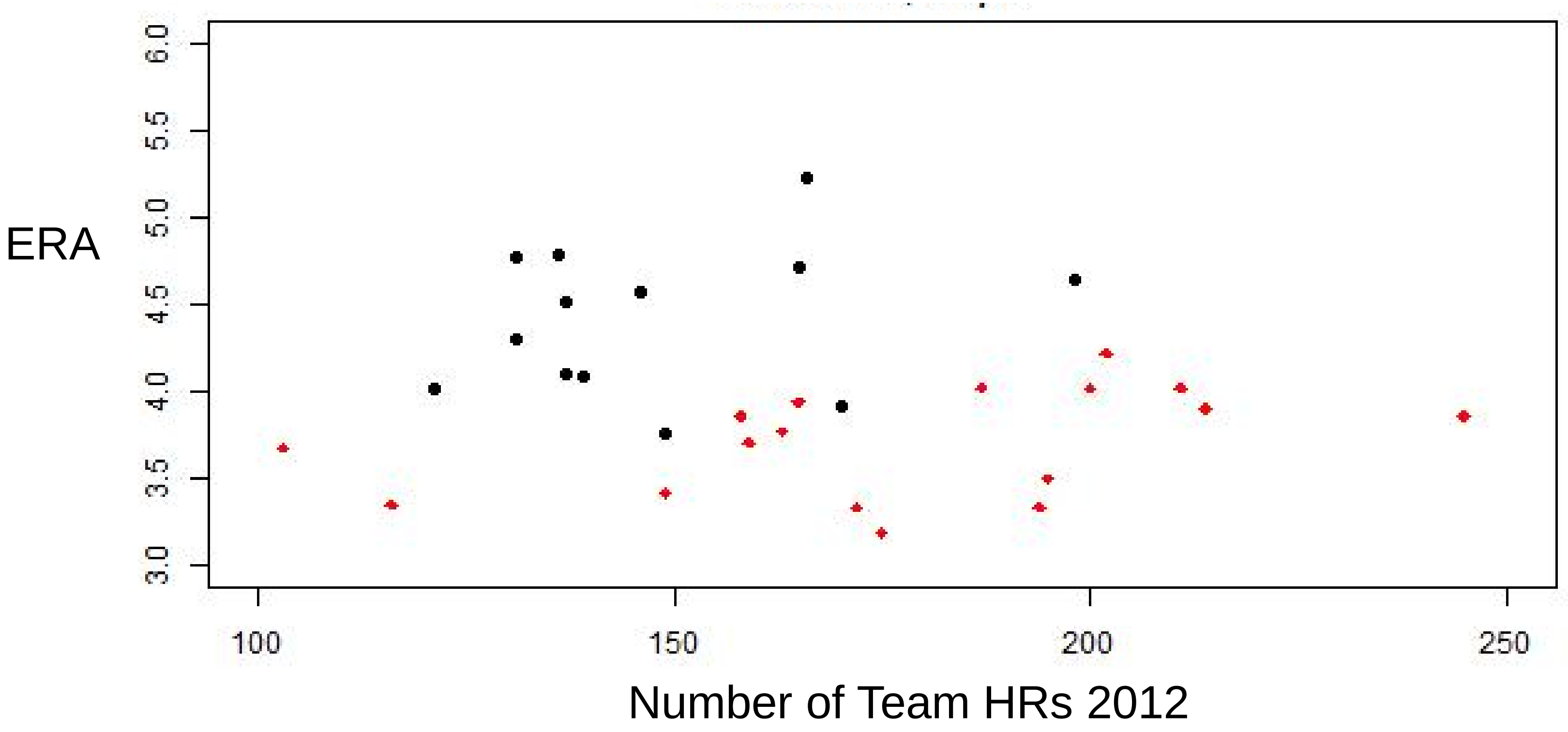


Number of Team HRs 2012 (call it '*x*')

A Model for Classification

- **2 classes: +1=Black (WL%>=.5) -1=Red (WL%<.5)**

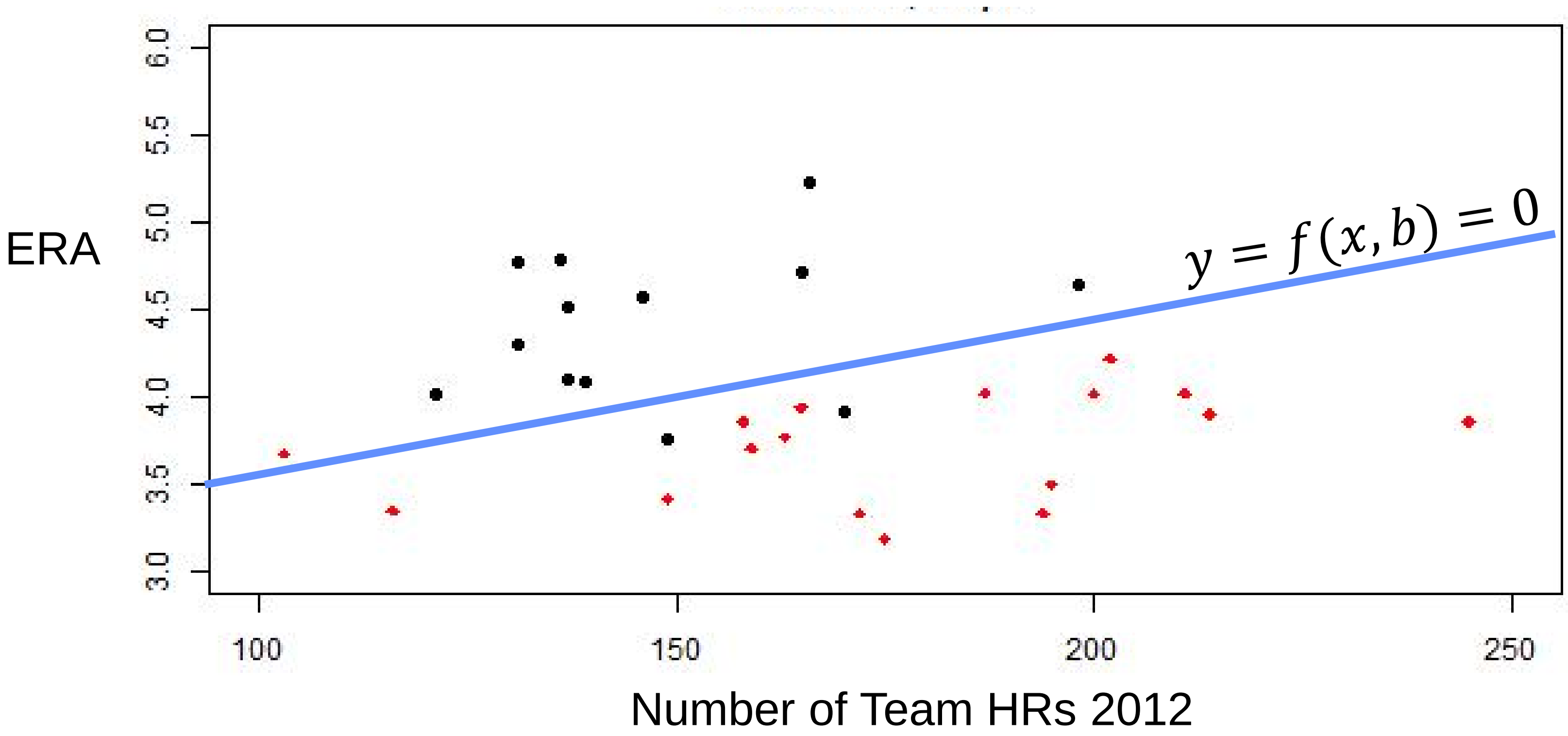
Q: Classify winning records based on HRs and ERA?



A Model for Classification

- **2 classes: +1=Black (WL%>=.5) -1=Red (WL%<.5)**

Q: Classify winning records based on HRs and ERA?

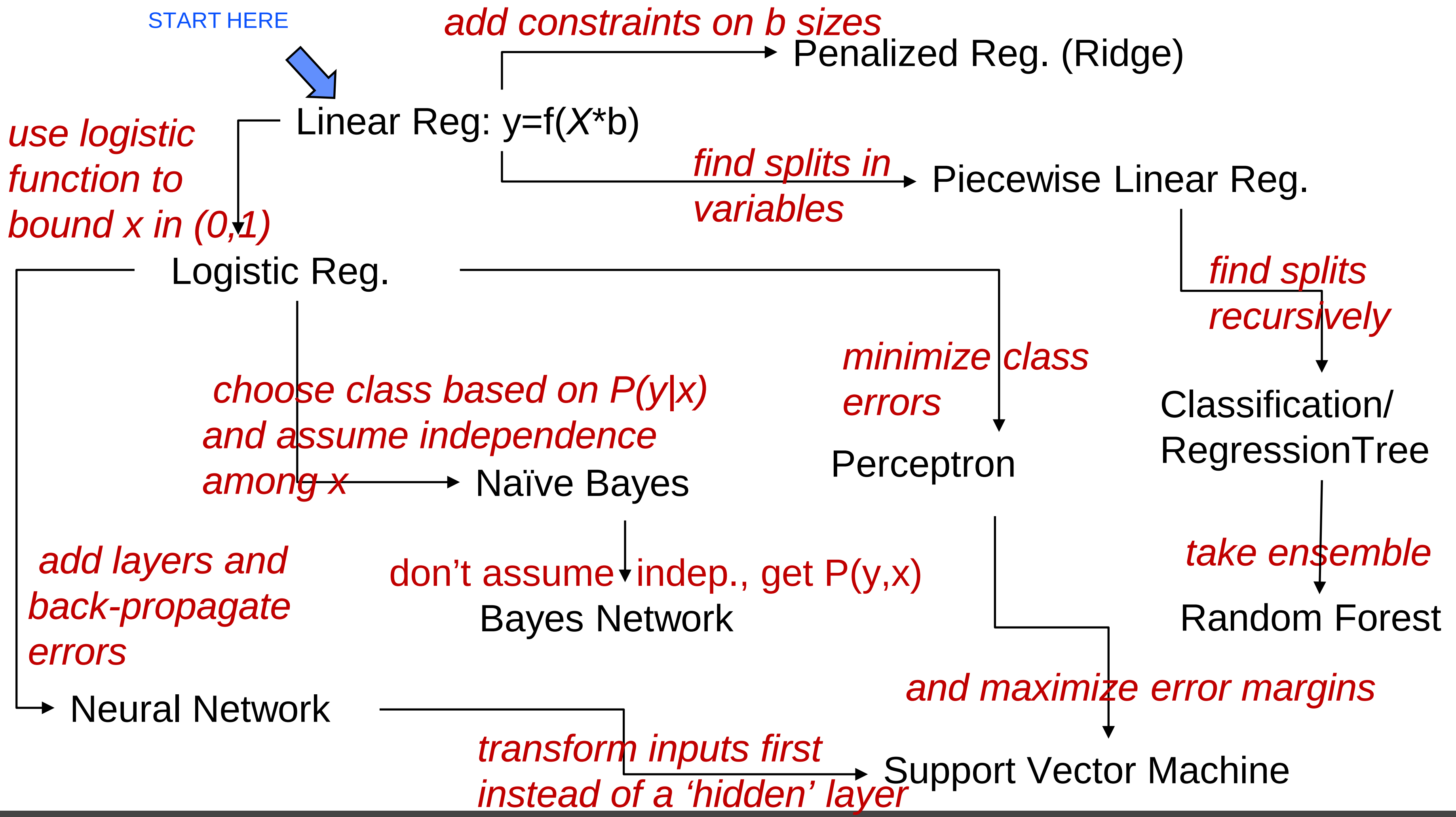


Fit parameters so that the decision threshold will minimize errors or maximize likelihoods

Model Choices

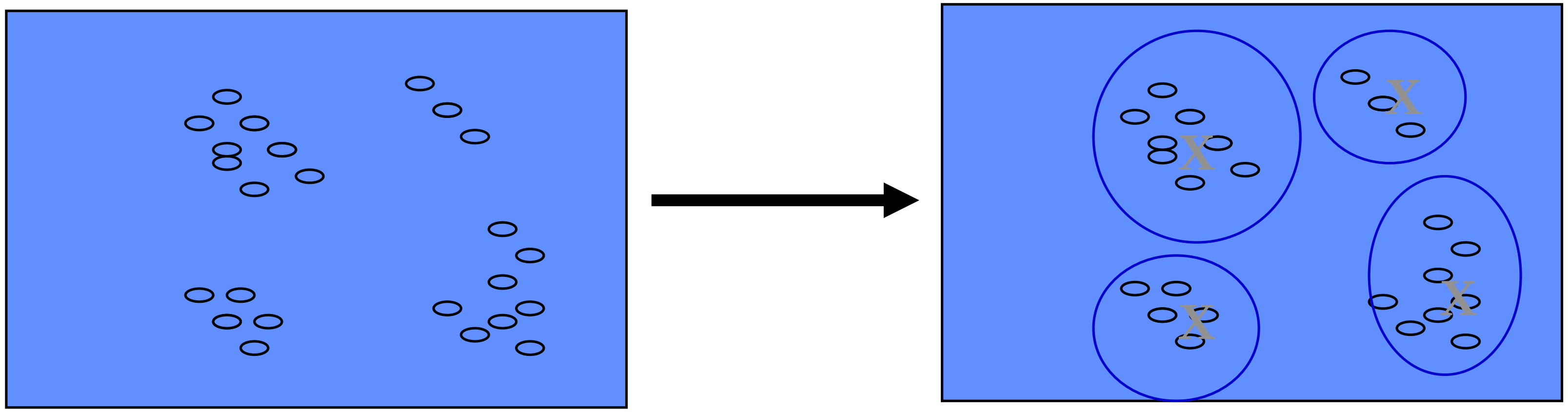
- **What kinds of functions to use**
  - e.g. Linear vs NonLinear
- **What to Optimize**
  - Minimize Sum Squared Error
  - Minimize Classification Errors
  - Maximize Probabilities
- **How to Fit Parameters**
  - Analytically
  - Search parameter space or follow gradients
  - Use Constraints as needed
- **How to avoid overfitting, and generalize well to test data**

Model Space Map – in a nutshell



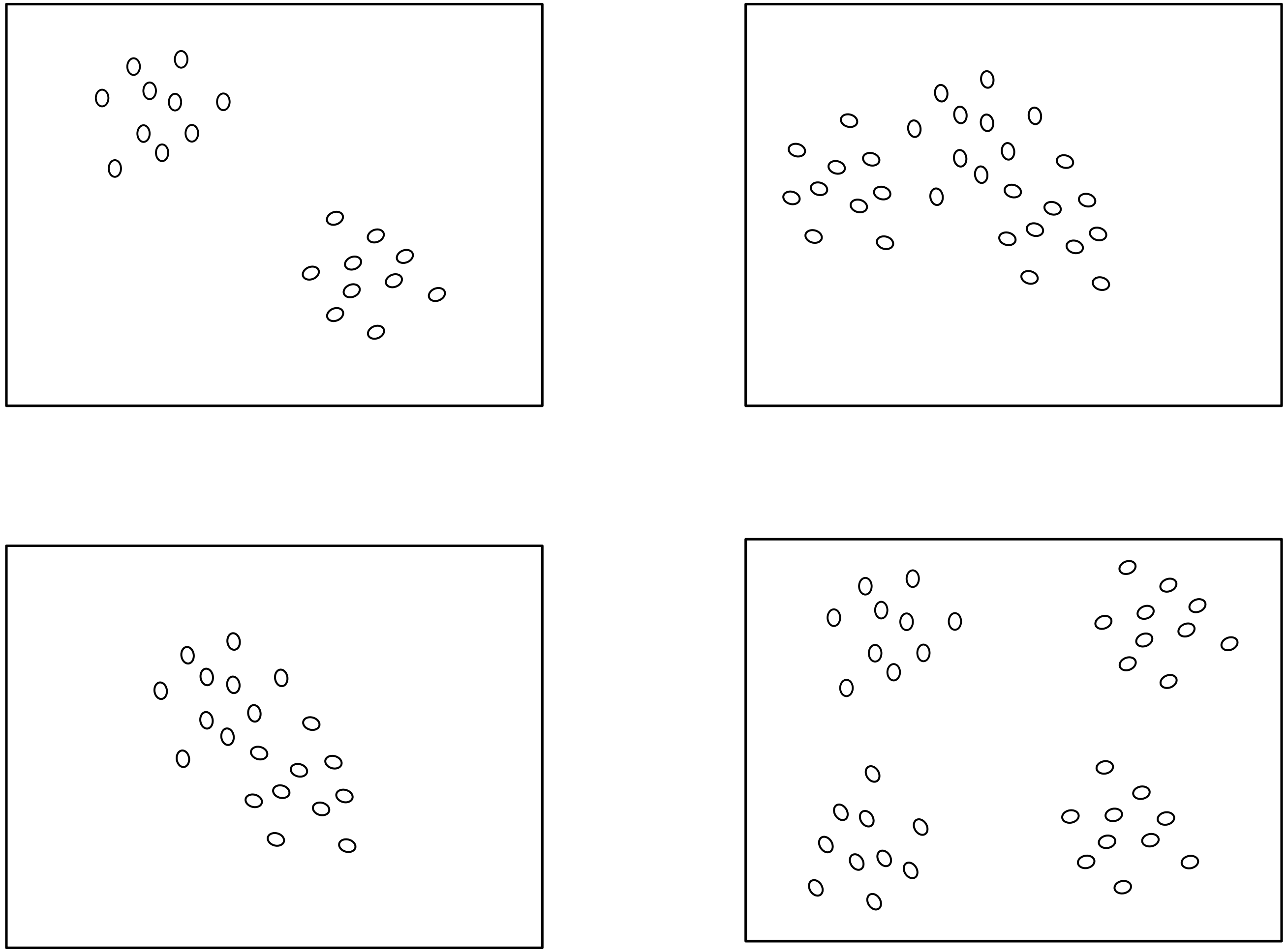
Clustering

- Basic idea: Group similar things together
- K-means
  - Partitioning instances into *k* disjoint clusters
  - Measure of similarity

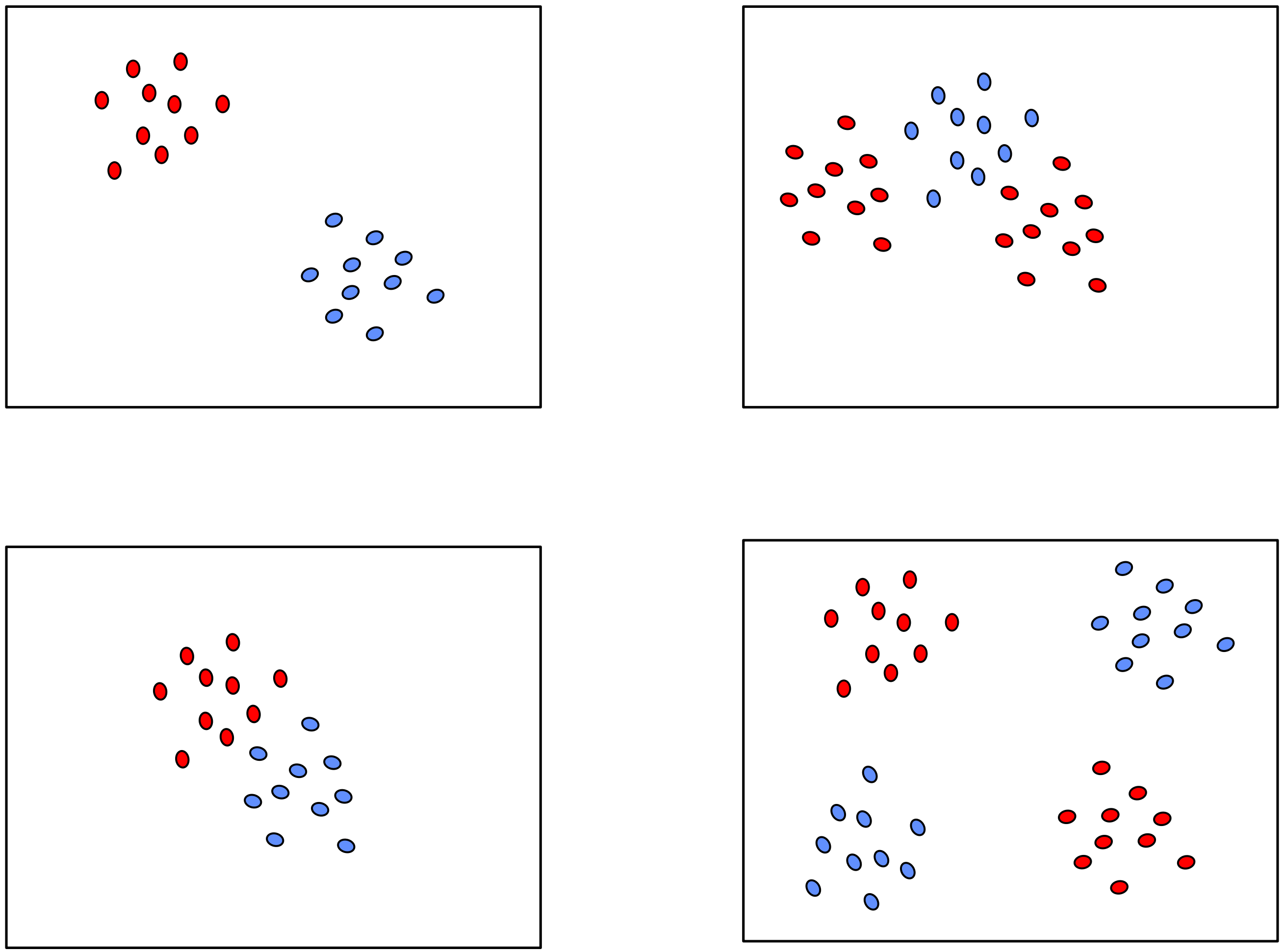


A note about clustering

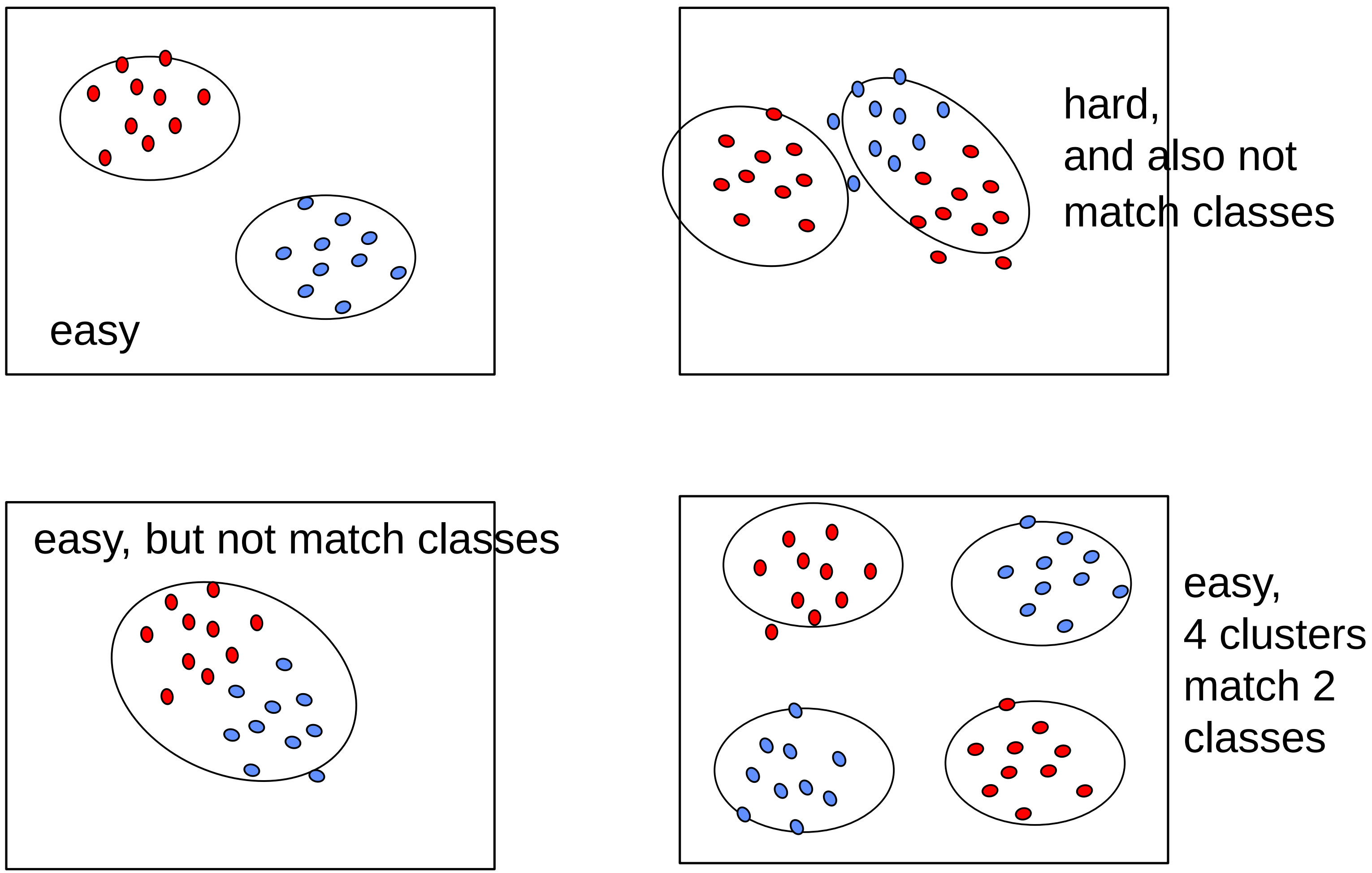
Imagine these 2 dimensional input spaces:
Which of these is easy or hard to cluster?
(e.g. separate into groups)



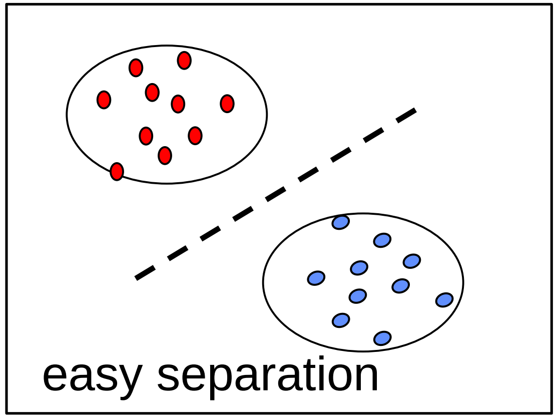
Now imaging there are two classes

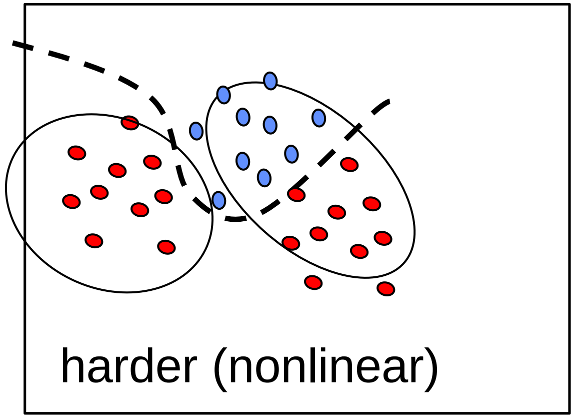


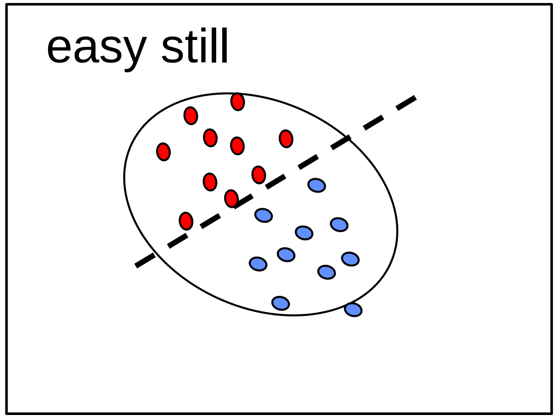
Potential clusters

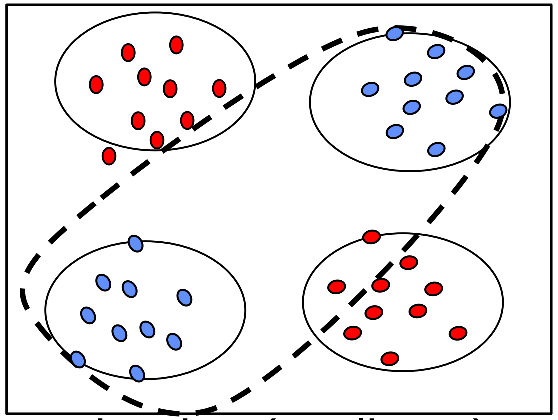


Which are easy or hard to classify? (ie separate red or blue with lines)









harder (nonlinear)

Upshot:
No easy relationship between clusters and classification

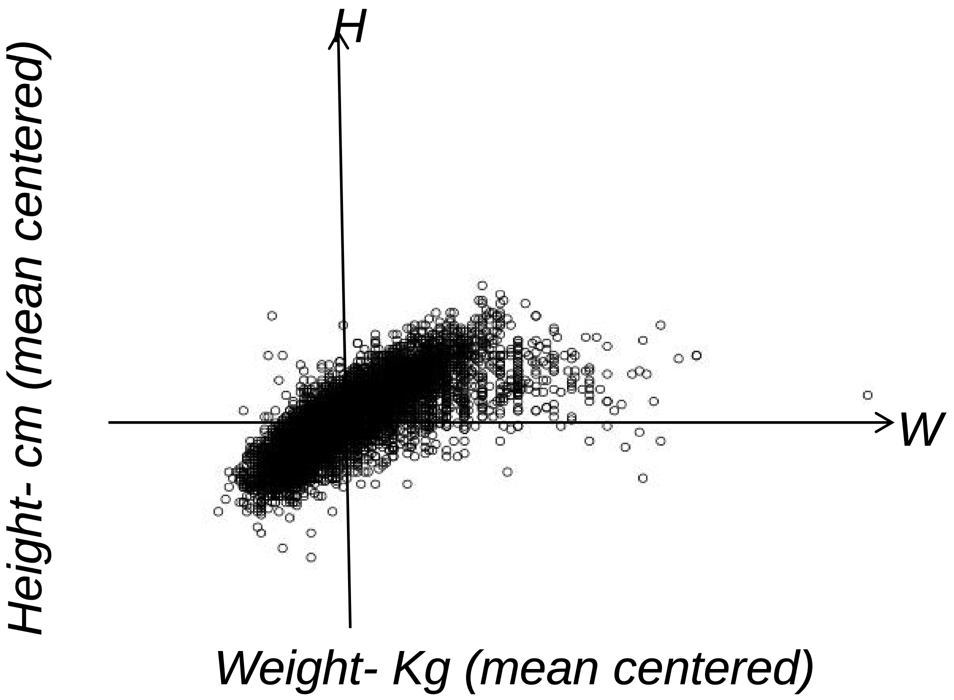
Matrix Factorization:

Given a numeric matrix, can we reduce the number of columns?

- Yes, if features are constant or redundant
- Yes, if features only contribute noise

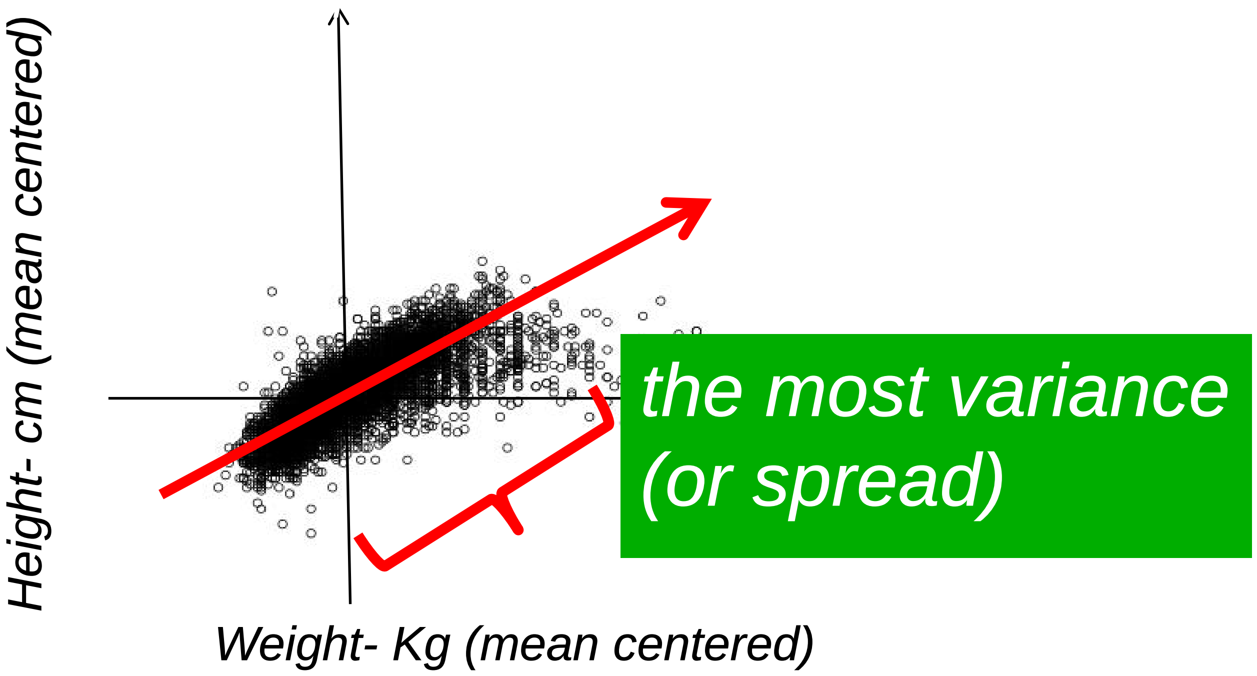
Conversely, want features that contribute to variations of the data

Example: Athletes’ Height by Weight

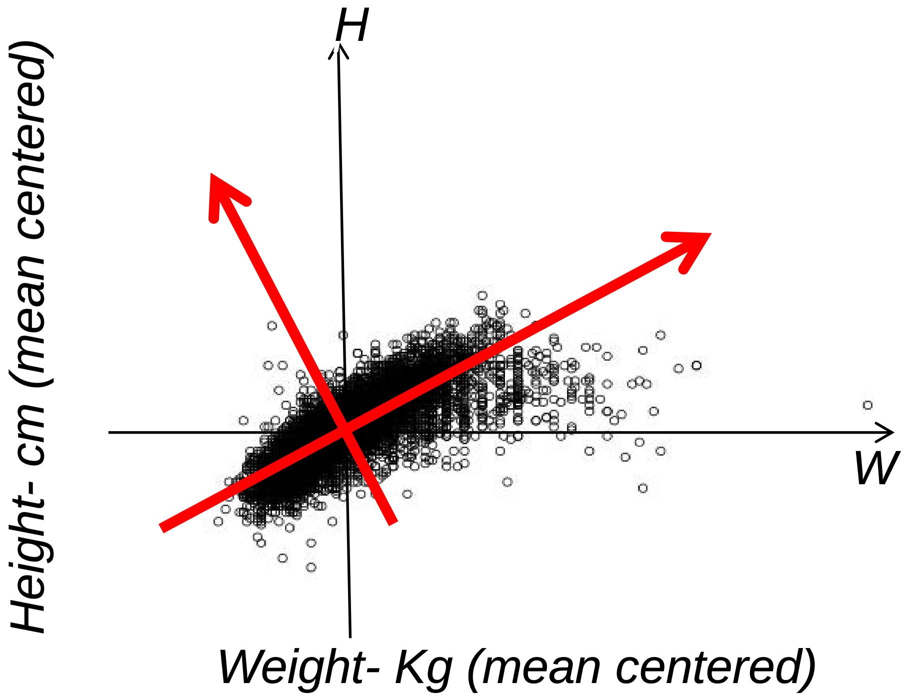


Find a line that aligns with the data.

Example: Athletes' Height by Weight

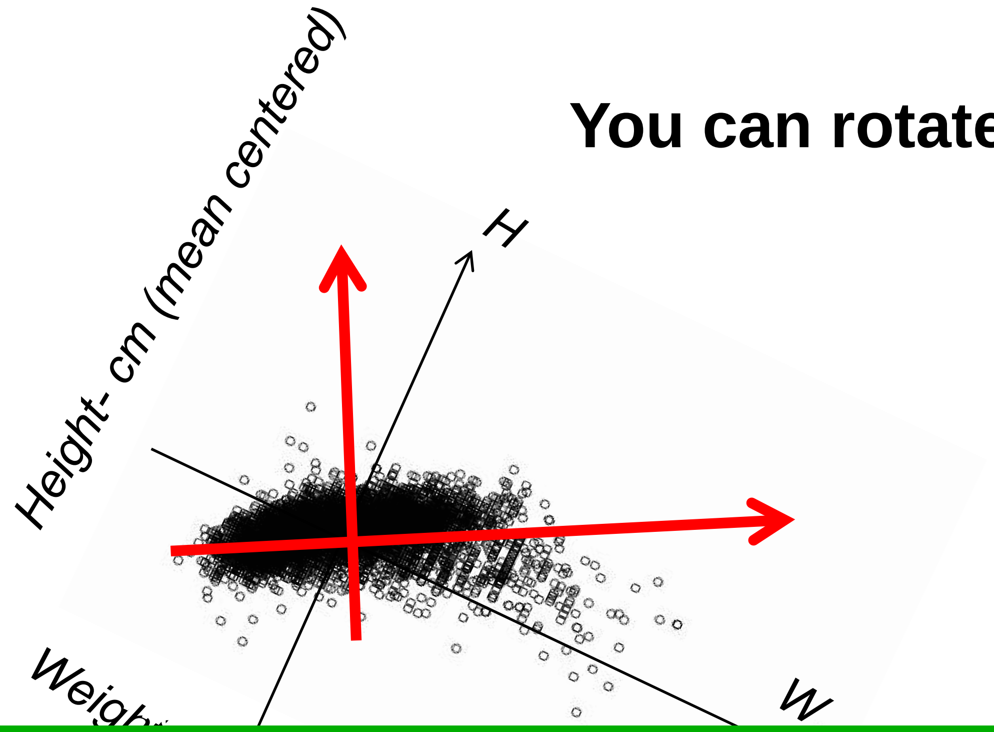


Find a line that aligns with the data.



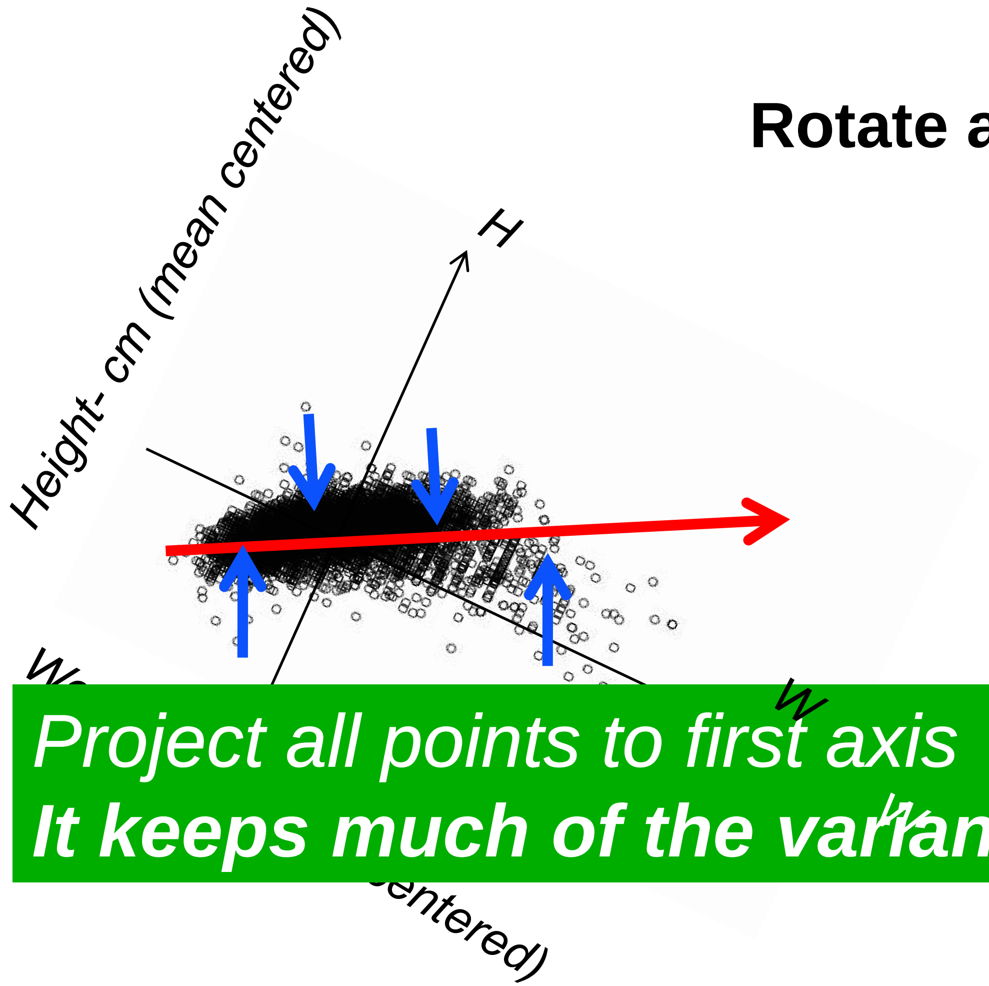
The next direction of most variance.

You can rotate the axes



New axes (i.e., new features or latent factors) are combinations of old axis (i.e., old features or observed factors)

Rotate axis



Project all points to first axis
It keeps much of the variance

Principle Components

- Best Known Factorization Algorithms:
  SVD (singular value decomposition)
  PCA (principle component analysis)

  SVD more generally works on non square matrices

- Can choose *k* factors heuristically as approximation improves, or choose *k* so that high percent (ie 80-95%) of data variance accounted for
- For higher dimensional data, use factors to visualize data in some 2D subspace

Exploration and Modeling Recommendations

- Start simple
- Consider trade off as you go more complex
- Find what works in your domain
- Find what works for this model

- **R, Python, Matlab: scripting languages with train/predict/test functions**
- **Weka, KNIME: GUI tools**

Pause

R, Scaling R, Parallel R

- A Glimpse of R
- R and Scaling
- Parallel options for R
- R on Comet exercise

The What and Why of R

- **A statistical computing environment**
  - Full set of Statistical/Mathematical functions
  - Programming Language for complete data manipulation
- **Free, Open Source**
- **Extended with user written packages**
- **Widely used in academic and increasingly in industry**

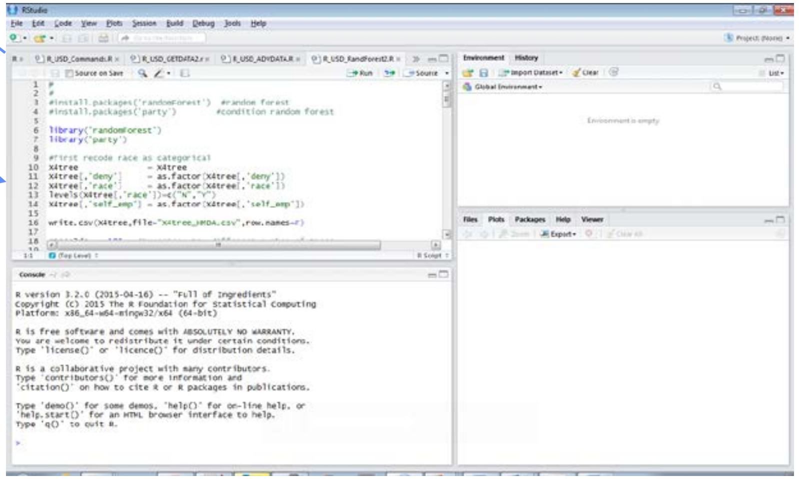
A typical R development workflow

- **R studio: An Integrated development environment for R on your local machine – good for development**

Menu tab

Edit window to
Build scripts

R console



Environment
Information on
variables and
command history

Plots, help
docs,
package lists

R commands in brief

- A typical R code workflow:

```
#READ DATA (housing mortage cases)
X               =read.csv('hmda_aer.csv',header=T,stringsAsFactors=T)
#SUBSET DATA
indices_2keep =which(X[,'s13'] %in% c(3,4,5)))
X               =X[unique(indices_2keep),]
#CREATE/TRANSFORM VARIABLES
pi_rat          = as.numeric(X[,'s46']/100)          #debt2income ratio
race            = as.numeric(X[,'s13'] %in% c(3,4)) #make race values 1-4 into values 0 or 1
deny            = as.numeric(X[,'s7']==3)            #make deny values into 0 or 1,
                                                        1 only for deny=‘3’
#RUN MODEL and SHOW RESULTS
lm_result       =lm(deny~race+pi_rat)                #lm is ‘linearmodel’
summary(lm_result)
```

R strengths for HPC

- **Sampling/bootstrap methods**
- **Data Wrangling**
- **Particular Statistical procedures that you won't find implemented anywhere else, e.g.**

  Multiple Imputation methods,

  Instrument Variable (2 stage) Regression

  Matching subjects for pairwise analysis

  MCMC routines

Scaling, practically

- **Scaling (with or without more data):**
  - more complex analysis (ie optimizations)
  - more sampling (ie more trees in Random Forest)

- **Sometimes easy to parallelize (like with sampling)**

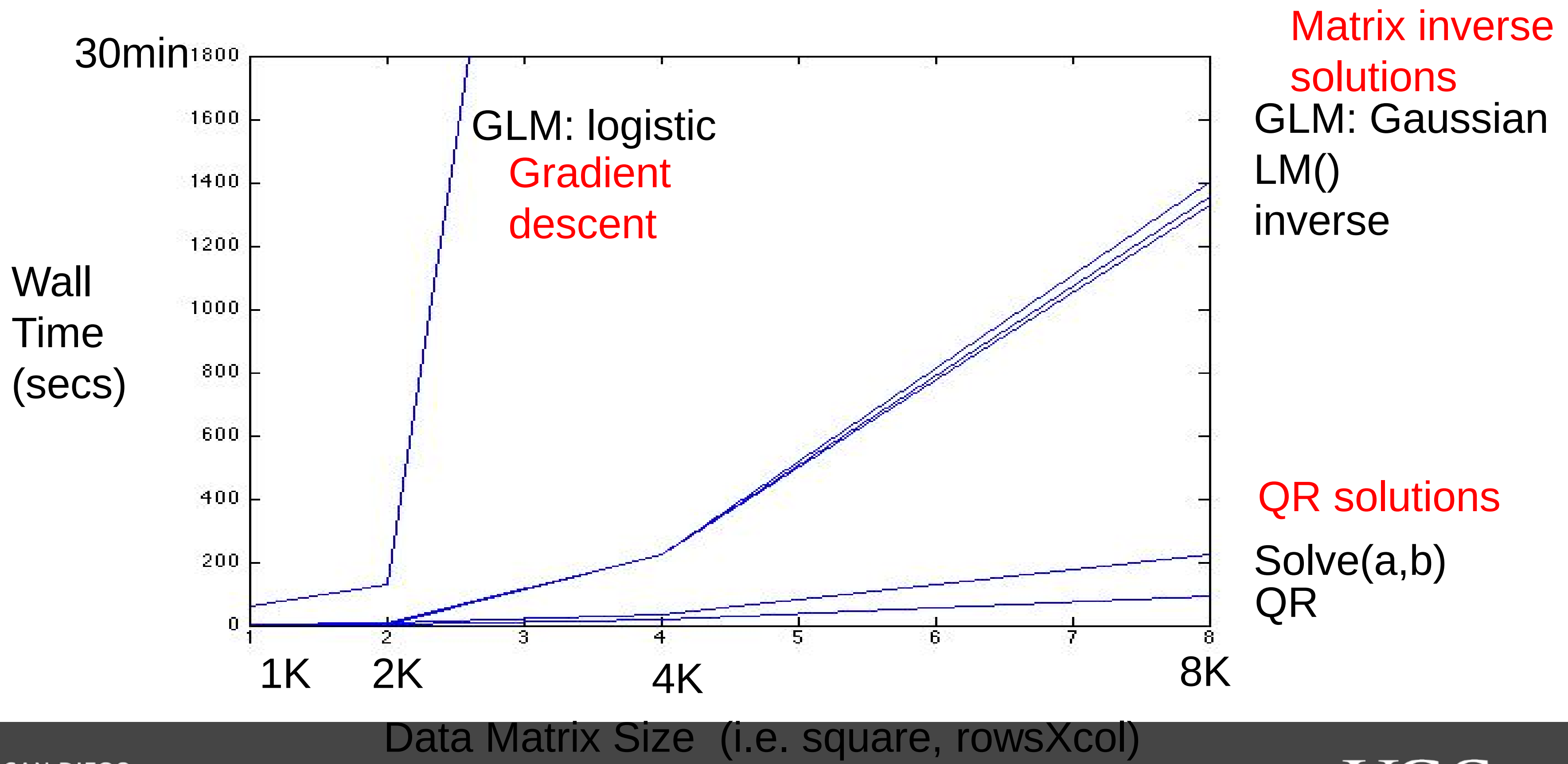
- **Sometimes too much communication between parts (matrix inversion)**

R Scaling In a nutshell

- R takes advantage of math libraries for vector operations
- R packages provide multicore, multimode, or distributed data (SparkR) options
- However, model implementations not necessarily built to use parallel backends
  - Some models more amenable to parallel versions

Solving Linear Systems Performance with R, 1 compute node

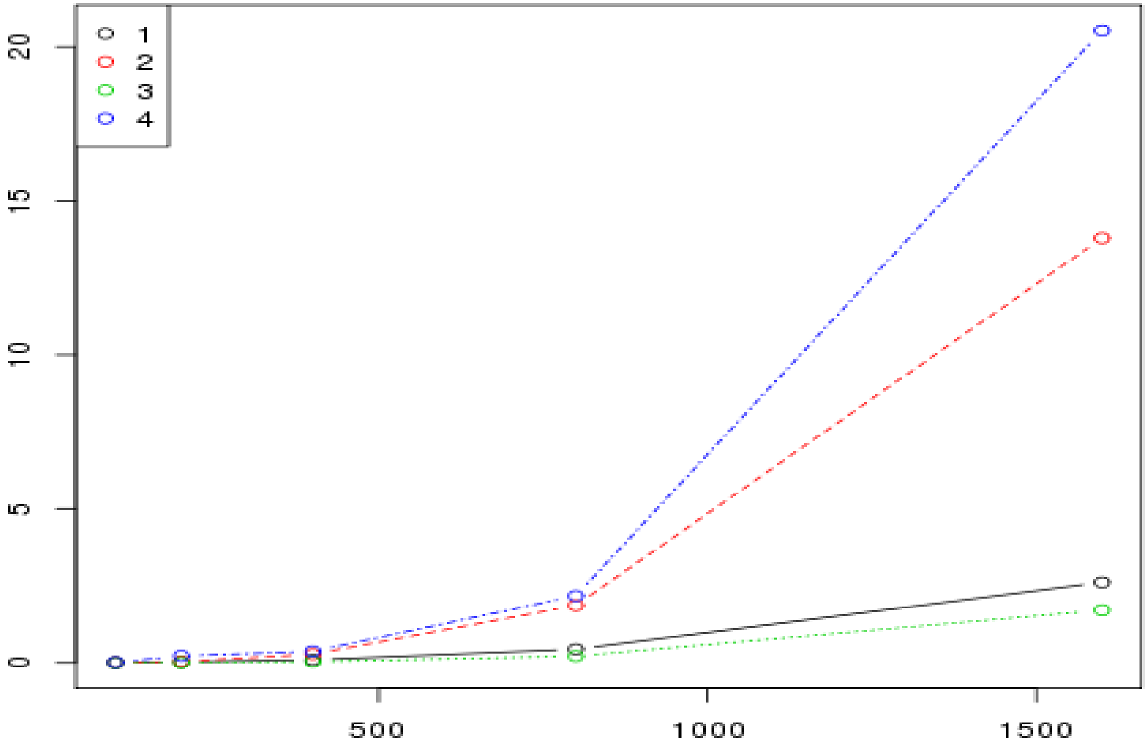
R:
glm(Y~X,family=gaussian) #gaussn regrssn (like lm)
glm(Y~X,family=binomial) # logistic regrssn (Y=0 or 1)



Machine learning models: Performance on 1 compute node

20 secs

Wall
Time
(secs)



Data Matrix Size (i.e. square, rowsXcol)

SVM classification (support vector machine optimization)

svd factoring

Kmeans

lm()

R multicore

- ‘doParallel’ package – provides the back end to the ‘for each’ parallel processing command
- uses threads across cpu cores to pass data & commands
- Updates and combines the previous ‘snow’ and ‘multicore’ packages, so that is also works for multinode.

See https://cran.r-project.org/web/packages/doParallel/vignettes/gettingstartedParallel.pdf

R multicore

- **Run loop iterations on separate cores**

```
install.packages(doParallel)
library(doParallel)
registerDoParallel(cores=24)
```

allocate workers

R multicore

- **Run loop iterations on separate cores**

```
install.packages(doParallel)
library(doParallel)
registerDoParallel(cores=24)

        my_data_frame = …..

        my_results = foreach(i=1:24,.combine=rbind) %dopar%
          {   …
                   your code here

              return(  a variable or object )
        })
```

allocate workers

%dopar% puts loops
across cores,
(loops are independent)
%do% runs it serially

R multicore

- **Run loop iterations on separate cores**

```
install.packages(doParallel)
library(doParallel)
registerDoParallel(cores=24)

      my_data_frame = …..

      my_results = foreach(i=1:24,.combine=rbind) %dopar%
        {   …
                your code here

            return(  a variable or object )
      })
```

allocate workers

%dopar% puts loops across cores,
(loops are independent)
%do% runs it serially

returned items 'combined' into list by default

specify to combine results into array with row bind

R multicore

- **Run loop iterations on separate cores**

BEWARE:
foreach will
copy data it
thinks is need to
every core

```
install.packages(doParallel)
library(doParallel)
registerDoParallel(cores=24)

   my_data_frame = …..

   my_results = foreach(i=1:24,.combine=rbind) %dopar%
     {   …
           your code here

         return(  a variable or object )
   })
```

allocate workers

%dopar% puts loops
across cores,
(loops are independent)
%do% runs it serially

specify to combine results into
array with row bind

returned [illegible]ems
‘combined [illegible]t
by default

R multinode: parallel backend

- **Run loop iterations on separate nodes**

```
library(doParallel)

cl <- makeCluster(48)
registerDoParallel(cl)
```

allocate cluster as
parallel backend

R multinode: parallel backend

- **Run loop iterations on separate nodes**

```
library(doParallel)

cl <- makeCluster(48)
registerDoParallel(cl)

my_data_frame = …..

results = foreach(i=1:48,.combine=rbind) %dopar%
   {   … your code here


        return(  a variable or object )
})
stopCluster(cl)
```

allocate cluster as parallel backend

%dopar% puts loops across cores and nodes

R multinode: parallel backend

- Run loop iterations on separate nodes

BEWARE: foreach will copy data it thinks is need to every node – that can take a long time!

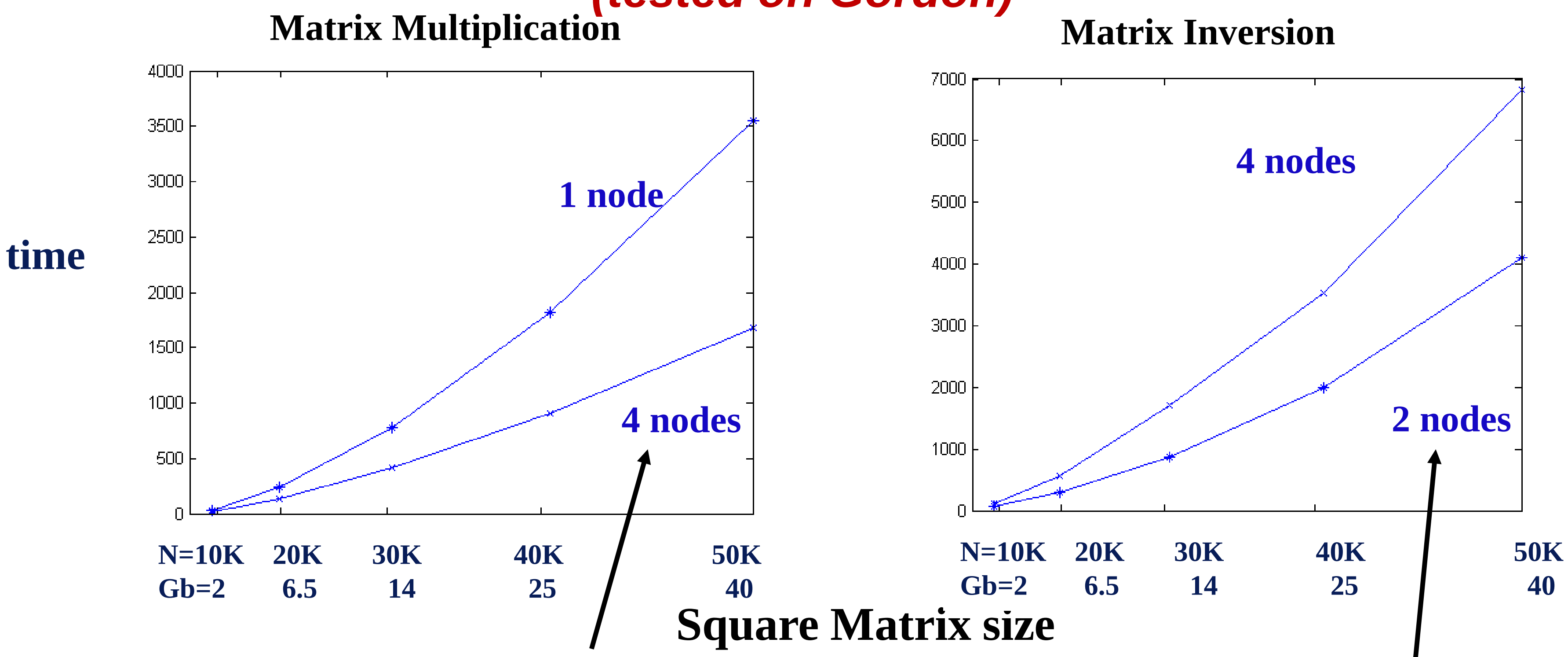
```
library(doParallel)

cl <- makeCluster(48)
registerDoParallel(cl)

my_data_frame = …..

results = foreach(i=1:48,.combine=rbind) %dopar%
   {   … your code here


       return(  a variable or object )
})
stopCluster(cl)
```

allocate cluster as parallel backend

%dopar% puts loops across cores and nodes

Multiple Compute Nodes not always help

(tested on Gordon)

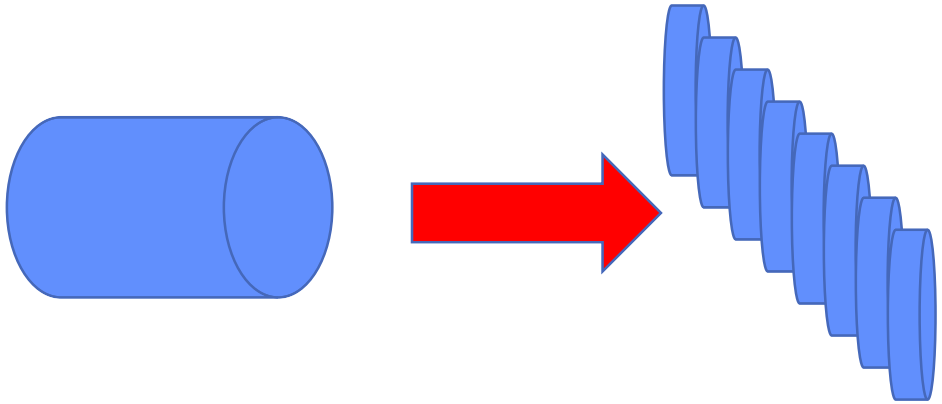


multinodes: more nodes is less time for multiplication,

less nodes is better for inversion

Another option for (embarrassingly) Parallel R

1. Split up data into N parts



Another option for (embarrassingly) Parallel R

1. Split up data into N parts

2. In slurm batch script:

ibrun -np processors My-perl-script

My-perl-script:
get cpu-id &
pass it to R

Another option for (embarrassingly) Parallel R

1. Split up data into N parts

2. In slurm batch script:

ibrun -np processors My-perl-script

Launch MPI wraps around Perl & R script

My-perl-script:
get cpu-id &
pass it to R

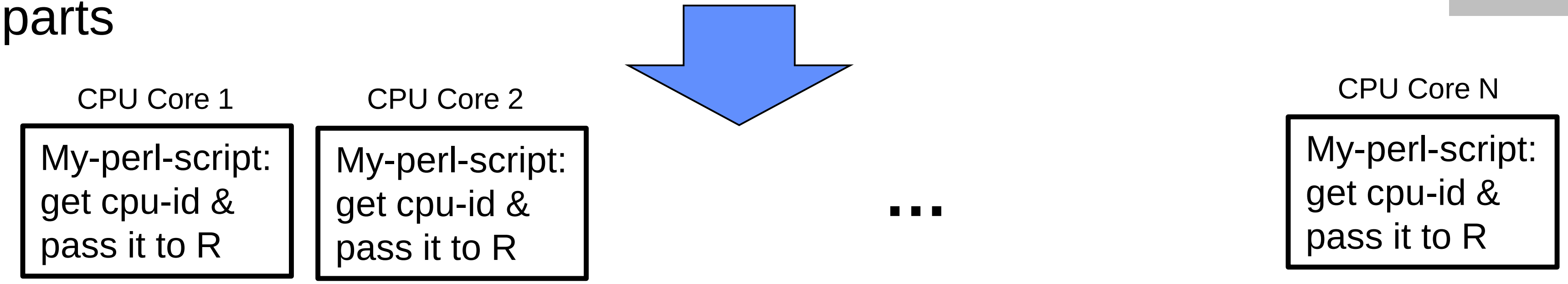
Another option for (embarrassingly) Parallel R

1. Split up data into N parts

2. In slurm batch script:

ibrun -np processors My-perl-script

Launch MPI wraps around Perl & R script



Another option for (embarrassingly) Parallel R

1. Split up data into N parts

2. In slurm batch script:

ibrun -np processors My-perl-script

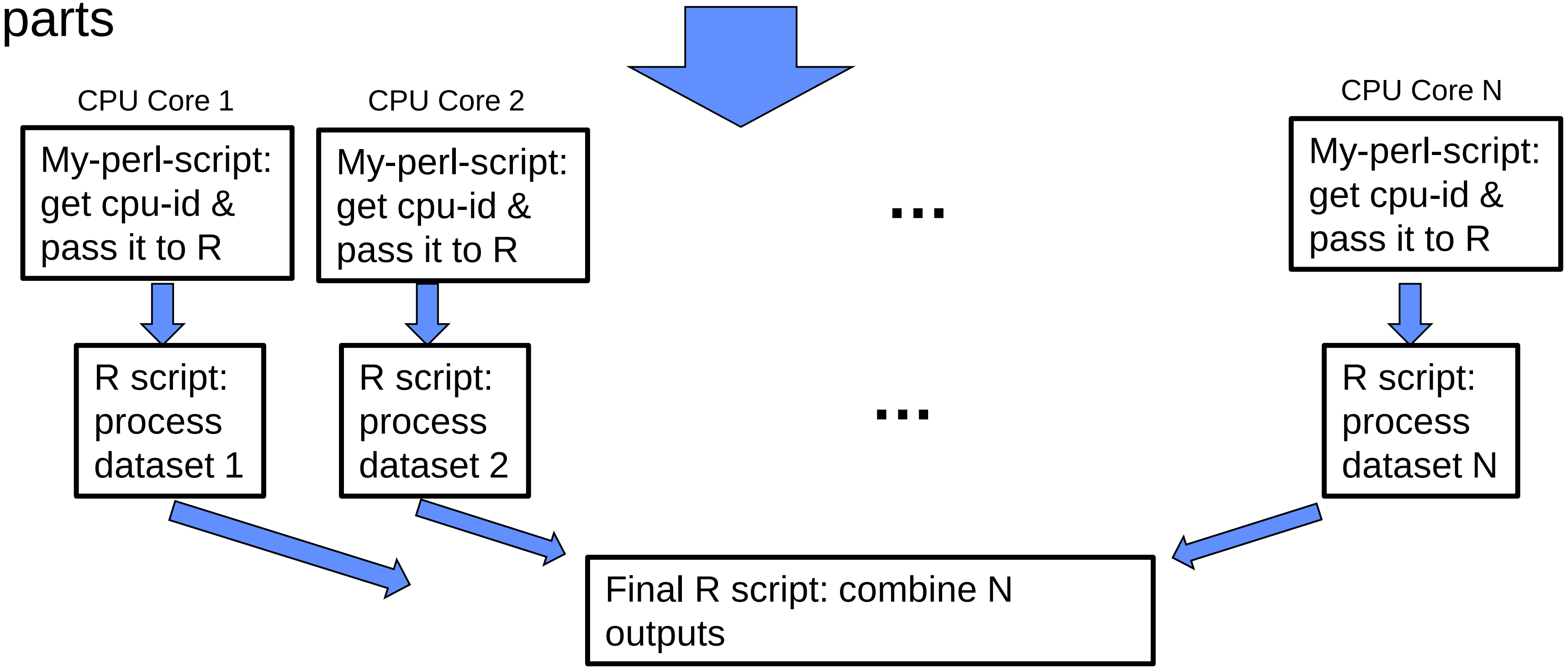
CPU Core 1	CPU Core 2		CPU Core N
My-perl-script: get cpu-id & pass it to R	My-perl-script: get cpu-id & pass it to R	…	My-perl-script: get cpu-id & pass it to R
R script: process dataset 1	R script: process dataset 2	…	R script: process dataset N

Another option for (embarrassingly) Parallel R

1. Split up data into N parts

2. In slurm batch script:

ibrun -np processors My-perl-script



Another option for (embarrassingly) Parallel R

1. Split up data into N parts

2. In slurm batch script:

ibrun -np processors My-perl-script

CPU Core 1

My-perl-script: get cpu-id & pass it to R

R script: process dataset 1

CPU Core 2

My-perl-script: get cpu-id & pass it to R

R script: process dataset 2

…

…

CPU Core N

My-perl-script: get cpu-id & pass it to R

R script: process dataset N

Final R script: combine N outputs

More programming but more flexible

Normal batch job info

```
#!/bin/bash
# ----------------------------
# slurm script for a batch job on comet
# to run a task on individual cores
# ----------------------------
#SBATCH --job-name="packR"
#SBATCH --output="serial-pack.%j.%N.out"
#SBATCH --partition=compute
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=24
#SBATCH --export=ALL
#SBATCH -t 1:00:00
#SBATCH -A sds164

bash

#Generate a hostfile from the slurm node list
export SLURM_NODEFILE=`generate_pbs_nodefile`
module load R

#launch 24x2=48 tasks on 48 cores,
# and start this perl script on each task
ibrun --npernode 24 --tpp 1 perl ./bundlerxP.pl

#One can also run hybrid:
#  launch 1 process per node, with 24 threads, and
#  use doParallel
ibrun --npernode 1 --tpp 24 perl ./bundlerxP.pl
```

ibrun the ‘bundler’ perl script on 24 cores per nodes, and 1 thread each

the
‘bundler
’
Perl
script

Get current
cpu id and
number of
processes

```
#!/usr/bin/perl
use strict;
use warnings;



my ($myid, $numprocs) = split(/\s+/, `./getid`);

# -------------------------------------
# launch an R session for this task
# -------------------------------------
my $task_index = $myid+1;
`module load R;/opt/R/bin/Rscript Test_PackingR.R
$task_index >
                         Rstd_out.$task_index.txt`;
```

the backtick
executes system
command

the rank id
as an
argument

Scaling doParallel vs ‘Packing’ R sessions

- **Packing *independent* R sessions onto cores is more flexible for:**
  - data management
  - large number of separate models
  - large variation in time per model
  - large matrix operations repeated
  - hybrid multimode/multicore scripts

But requires more programming or preprocessing

Example: scaling MCMC

Distributed Markov Chain Monte Carlo for Bayesian Hierarchical Models, Frederico Bumbaca, UCIrvine, et al in print

- Probabilities of user web activity interdependent through a hierarchical model
- MCMC search for probabilities made independent through a phased approach.
- Ran on SDSC Comet with **'serial packing'** parallelization

(Using rhierMnlRwMixturefunction in the R package, bayesm)

# Individuals	Cores	Individ per Core	Total Minutes (I/O time)
100 million	1,7282 (max)	~ 58K	206 (38)

Installing your own R Packages

- **In R:**

  install.packages('package-name')

  (see https://cran.r-project.org/ for package lists and reviews)

- **on Comet:**

  install.packages('ggmap',
  repos='http://cran.us.r- project.org',dependencies=TRUE)

  If compiling is required and you get an error, call support

Other R packages:

- **Rspark - R interface to Spark**
- **pdbR - higher level over R-MPI, distributed matrix support and other**

  (better for dense matrices vs Spark)

- **R openMP**

  (e.g. if you want to program your own foreach)
- **Ff, bigmemory – map data to files**

  (can help with foreach)

- **HiPLAR - GPU and multicore for linear algebra**
- **Rgputools – GPU support**

pause

R on Comet terminal window

1. Get a compute node:

```
[Unix]$ :   srun --partition=debug --pty --nodes=1 --ntasks-per-node=24 -t 00:30:00
--wait=0 --export=ALL  -A your-account  /bin/bash
```

2. Start R

```
[Unix]$ module load R
[Unix]$ R              (this gets an interactive R session)


>quit()                (to exit R)

[Unix]$ exit           (to exit the compute node)
```

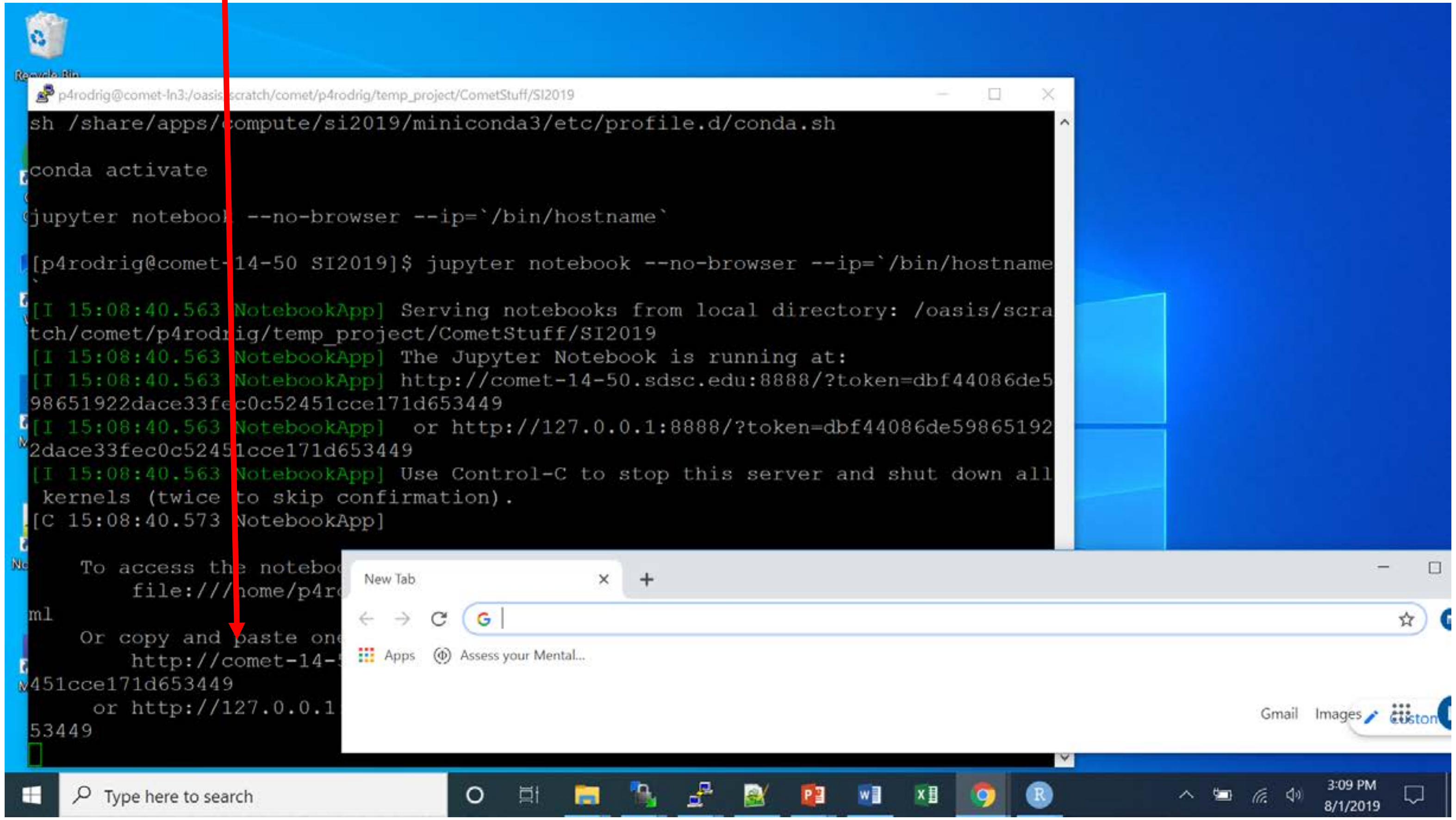
R multicore exercise

- **Login to comet**
  - cd to this lecture folder
- **Get an interactive compute node session**
- **Start notebook**
  - jupyter notebook --no-browser --ip="*" &

R parallel exercises

- **Open & run TestdoParallel Exercise 1,2,3**
  - remember that foreach assumes independence between loops
  - Start with smallish N,P
- **Look at memory usage in top command**
- **R does not well manage large data frames across cores**
  - N=800000 P=2000, makes ~12Gb data frames, R fails
- **Ex 3 will split up data for large data frames and have each core read a separate data**

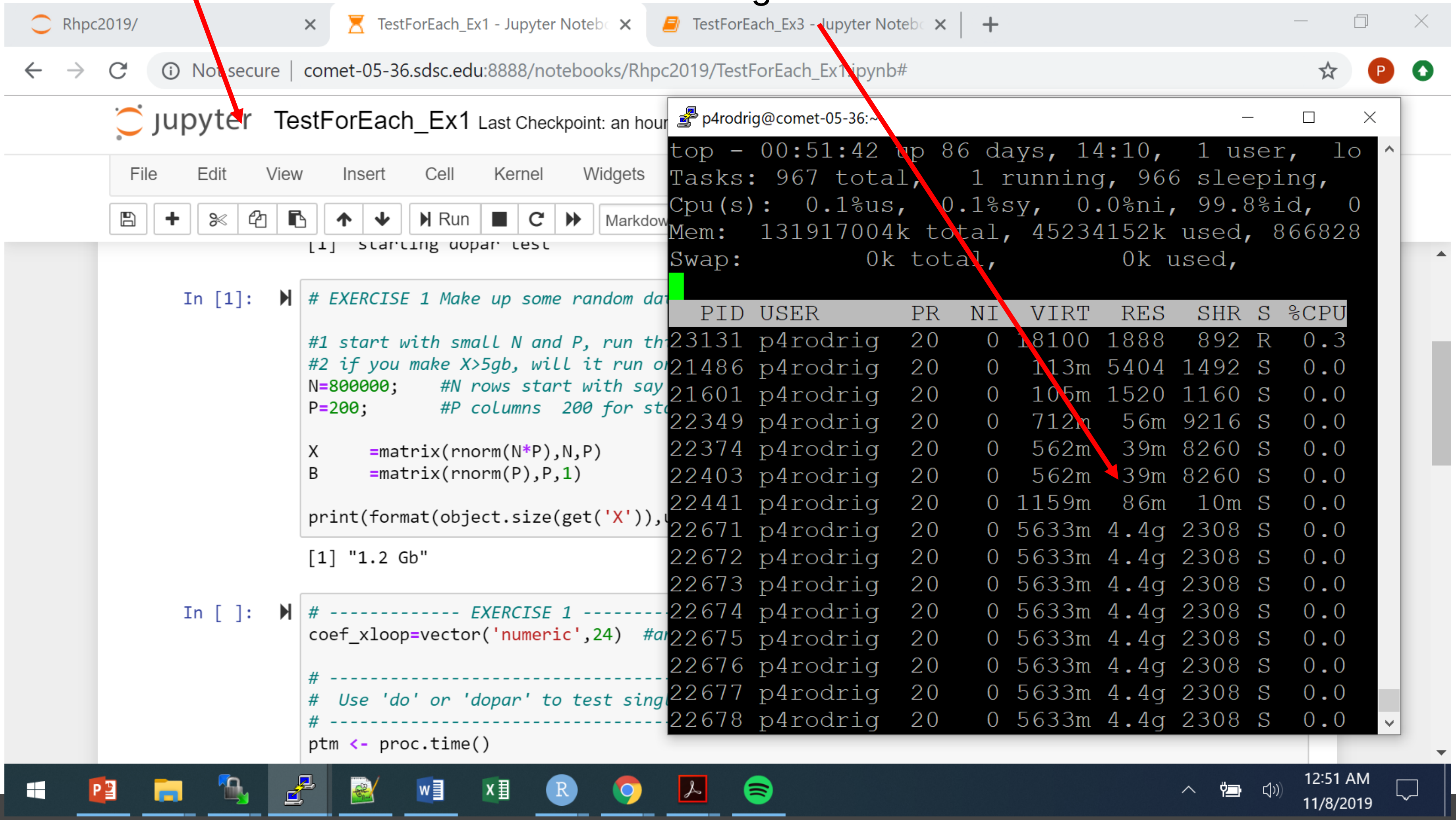
Starting jupyter notebook and copy paste URL into browser



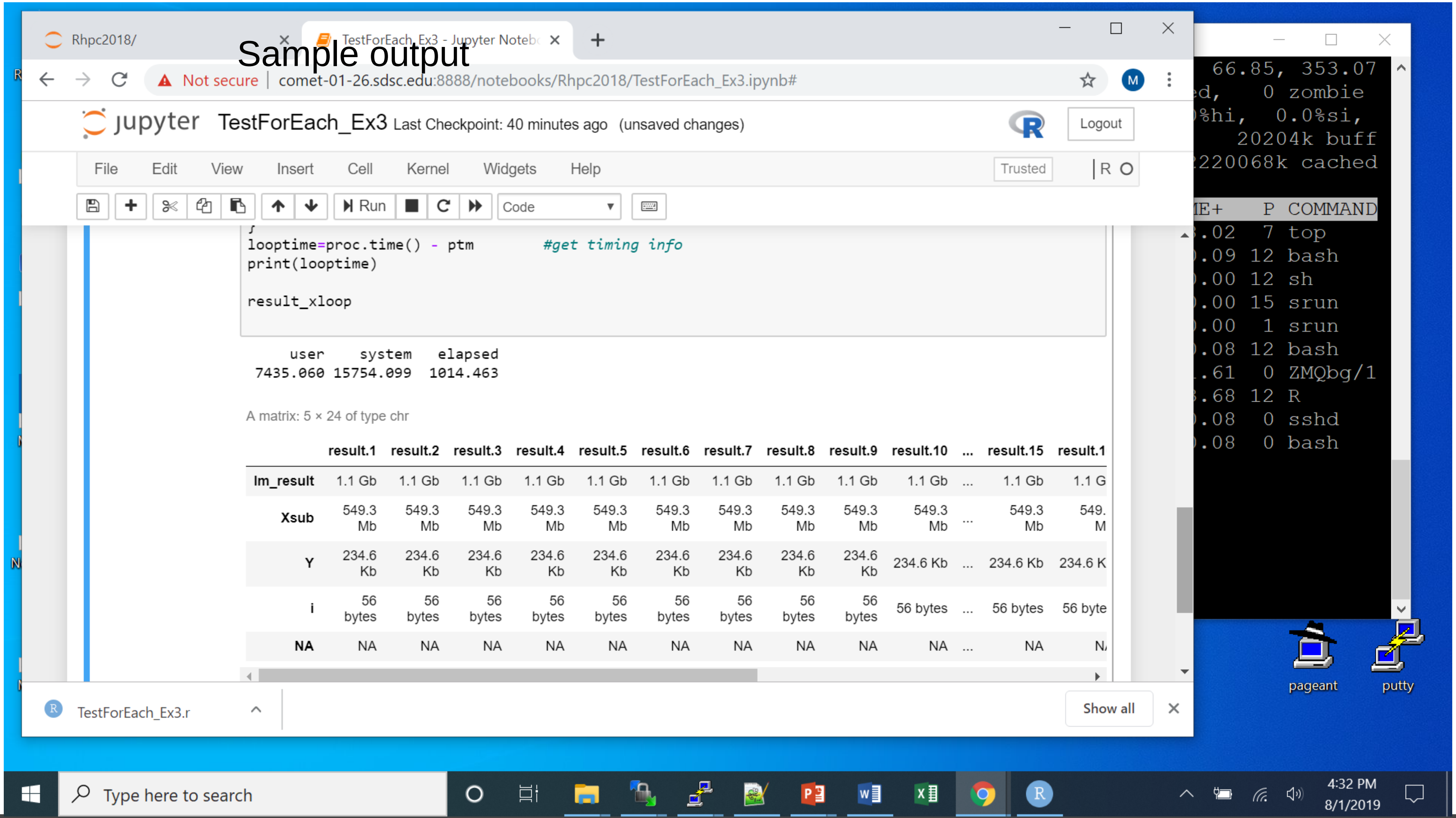
Select Rhpc2019 folder and select TestdoParallel exercises

Open 2nd terminal window directly in to comet-XX-XX.sdsc.edu compute node

Run top –u $user (then enter H) to see usage



Sample output



- **Pause**

pbdR package

- **API on top of MPI and Scalapack Lin. Algebra library**
- **Sets up virtual grid to handle large matrix multiplication**

See https://pbdr.org/packages.html

pbdR sample code

```
library(pbdDMAT)

init.grid()                # <<< ----  pbdR will select grid sizes for you by default

myr   =comm.rank()
mys   =comm.size()

#Simple ways to print information
comm.print(paste("comm print myrank:",myr, " size:",mys),all=FALSE)

p=10000
dx <- ddmatrix(rnorm(p*p*10),p*10,p)       # <<< --- “ddmatrix”  - options to indicate global matrix
comm.print(dx,all=F)                       dimension, local dimension, and blocking sizes

….
```

To run: edit Runpbd script and enter: sbatch Runpbd

Test 1

For 1 node 24 cores:

```
Using 6x4 for the default grid size

[1] "comm print myrank: 0  size: 24"
[1] " matrix width: 10000"
--------------------------------------------------------------------------
orterun noticed that process rank 0 with PID 26491 on
node comet-18-56 exited on signal 9 (Killed).
--------------------------------------------------------------------------
```

But runs out of memory
(2 nodes 24 cores also runs out of memory)

Test 2

For 1 node 12 cores:

data split up among cores

Runs in about 950 secs (for a matrix multiplication)

```
Using 4x3 for the default grid size

[1] "comm print myrank: 0  size: 12"
[1] " matrix width: 10000"
COMM.RANK = 0

DENSE DISTRIBUTED MATRIX
---------------------------
Process grid:                4x3
Global dimension:            100000x10000
(max) Local dimension:       25008x3344
Blocking:                    16x16
BLACS ICTXT:                 0
```

Test 3

For 2 node 12 cores:

Using 6x4 for the default grid size

[1] "comm print myrank: 0 size: 24"
[1] " matrix width: 10000"
COMM.RANK = 0

DENSE DISTRIBUTED MATRIX

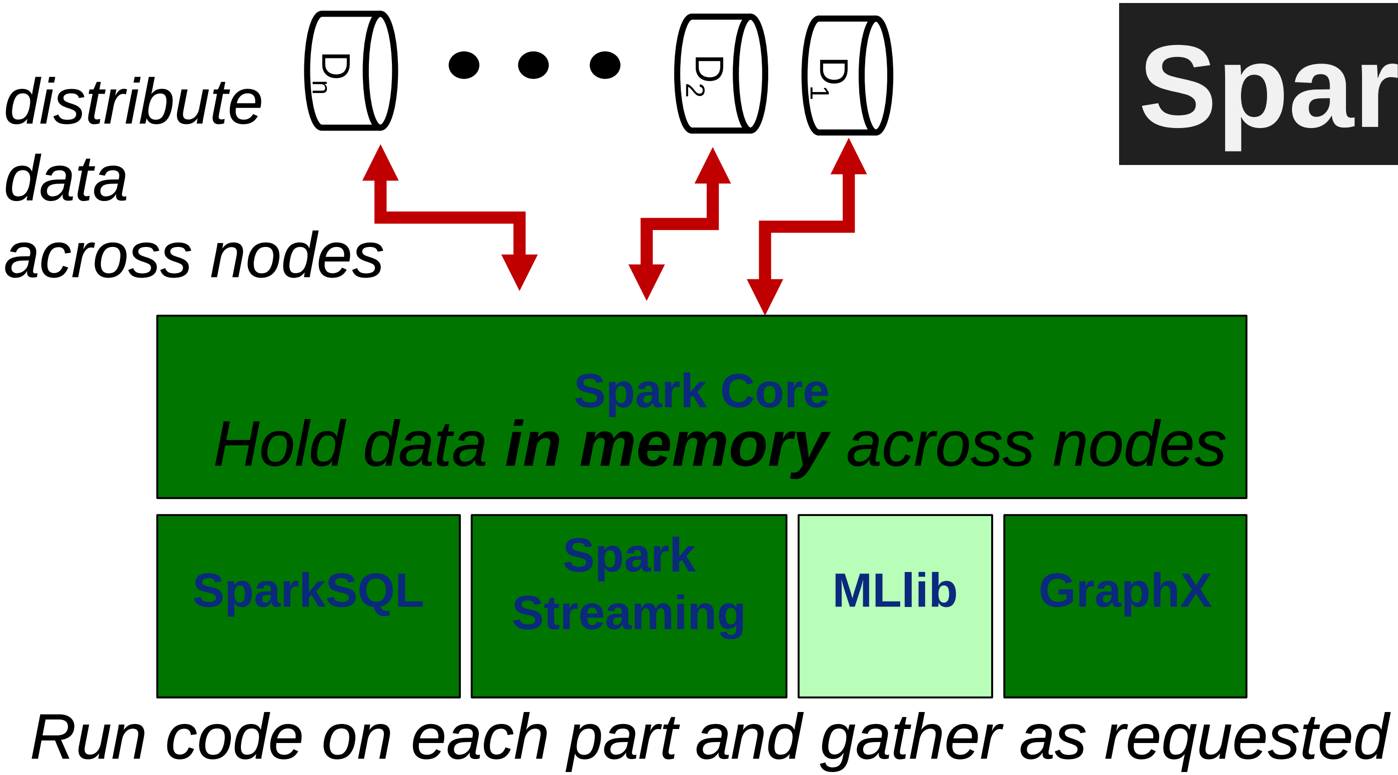
Process grid:	6x4
Global dimension:	100000x10000
(max) Local dimension:	16672x2512
Blocking:	16x16
BLACS ICTXT:	0

Runs in about 320 secs
(for a matrix multiplication)

Spark ML for bigger data:

- **Spark MLlib –**
  - Many standard Machine Learning models that are easiest to parallelize
    - Matrix Factorization
    - Naïve Bayes
    - Linear/NonLinear Regression Models with gradient descent optimization
    - Kmeans
  - Some support for large matrix operations

Spark MLlib



- Distributed implementations of common ML algorithms and utilities
- APIs for Scala, Java, Python, and R
- Scales well for independent processes

- **On to deep learning…**

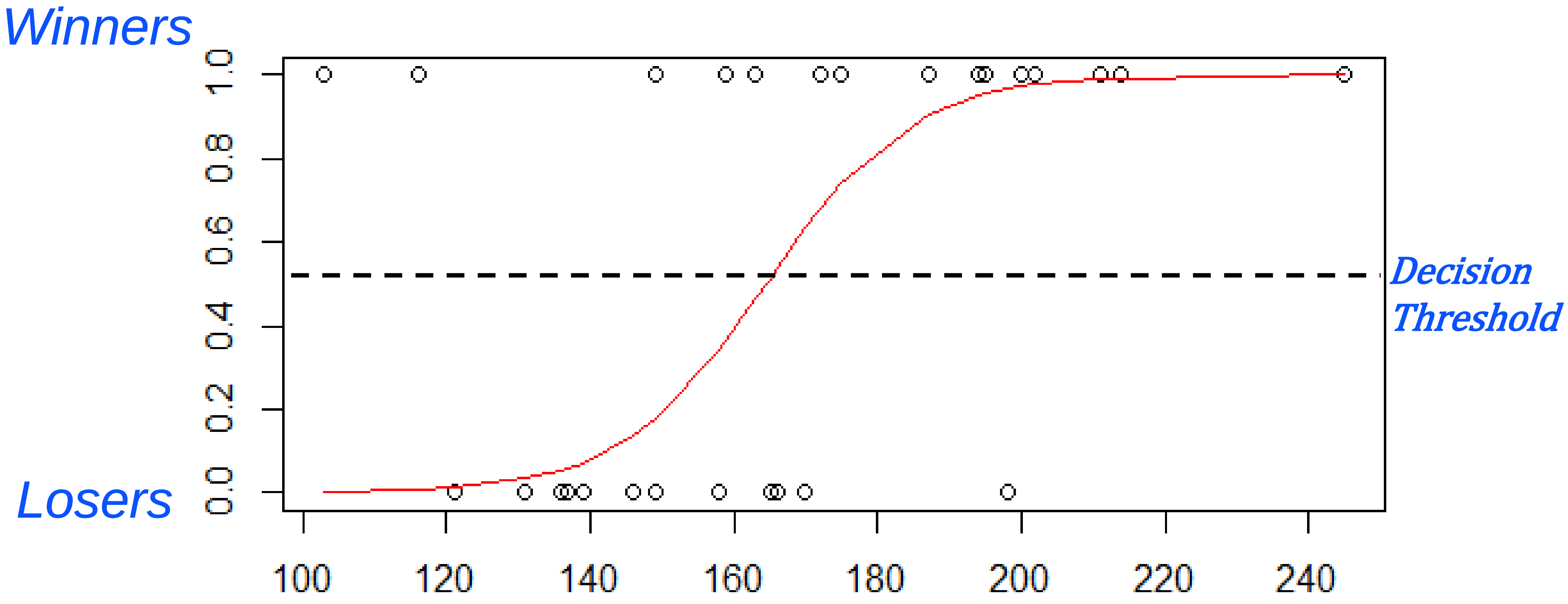
Deep Learning

- **3 characterizations:**

1. Learning complicated interactions about input
2. Discovering complex feature transformations
3. Using neural networks with many layers

Logistic regression

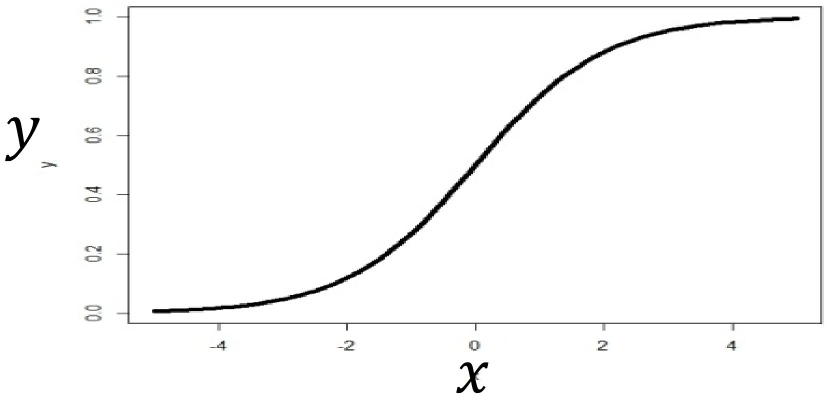
the Model: $y = f(x, b) = 1/(1 + \exp[-(b_o * 1 + b_1 * x)])$



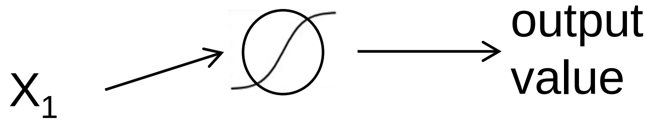
Number of Team HRs 2012 (call it '*x*')

Logistic to Neural Network model

- **In other words –**
  - Squash $(b_o * 1 + b_1 * x)$ to 0,1 range using logistic function

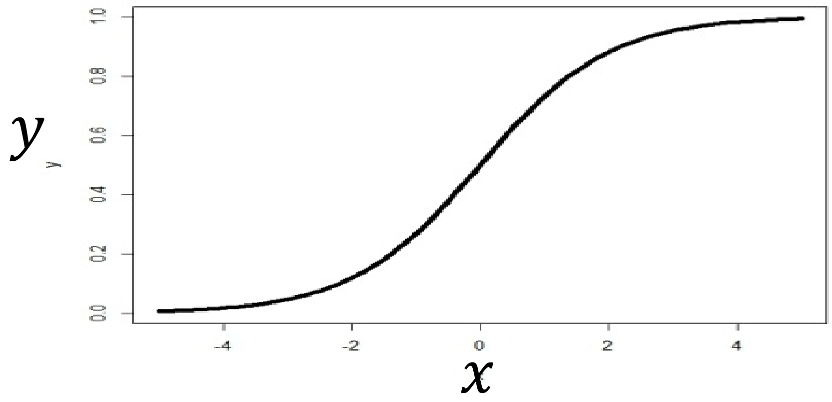


  - ***And use graphical depiction:***

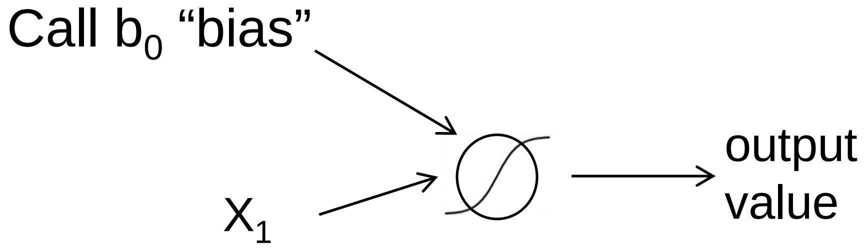


Logistic to Neural Network model

- **In other words –**
  - Squash $(b_o * 1 + b_1 * x)$ to 0,1 range using logistic function

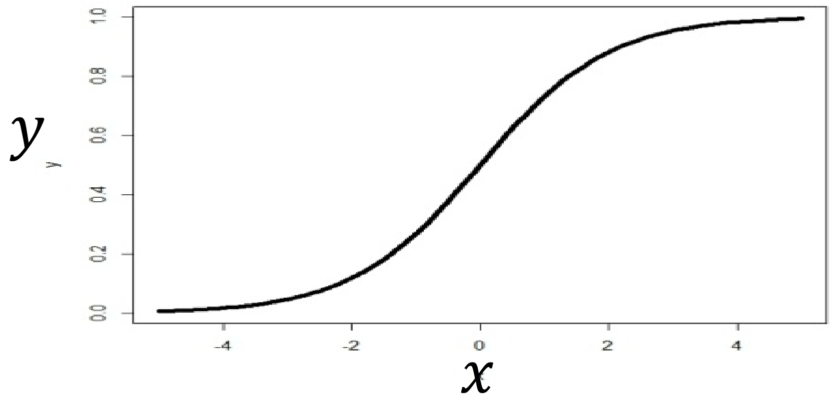


  - ***And use graphical depiction:***

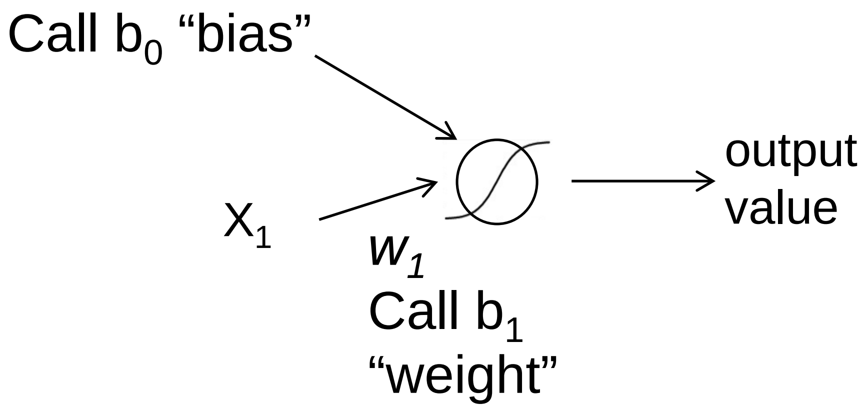


Logistic to Neural Network model

- **In other words –**
  - Squash $(b_o * 1 + b_1 * x)$ to 0,1 range using logistic function

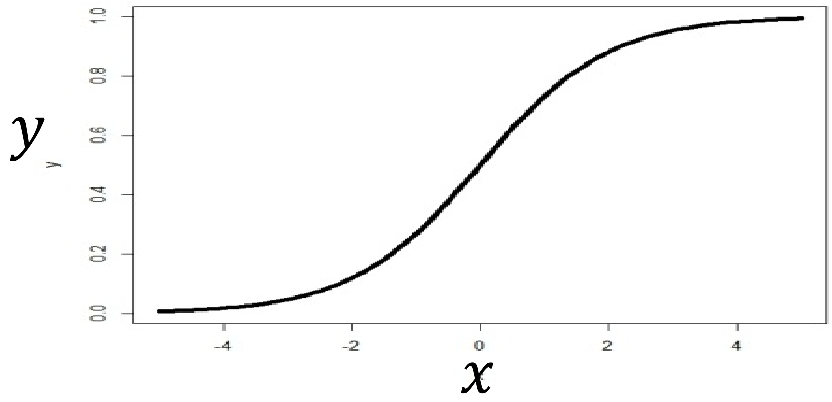


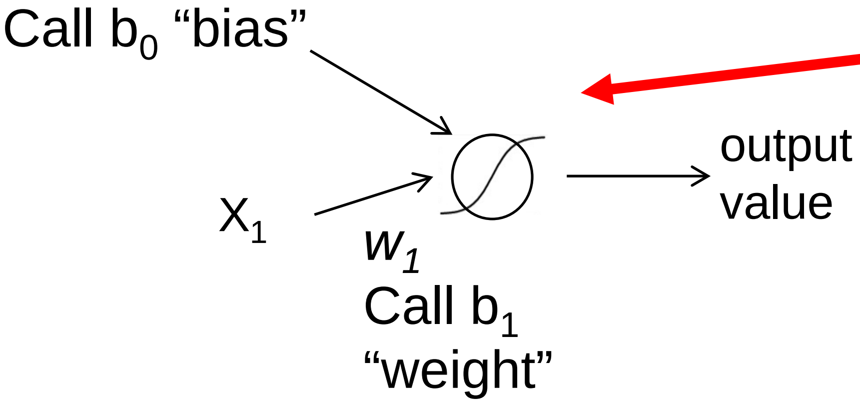
  - ***And use graphical depiction:***



Logistic to Neural Network model

- **In other words –**
  - Squash $(b_o * 1 + b_1 * x)$ to 0,1 range using logistic function
  - ***And use graphical depiction:***

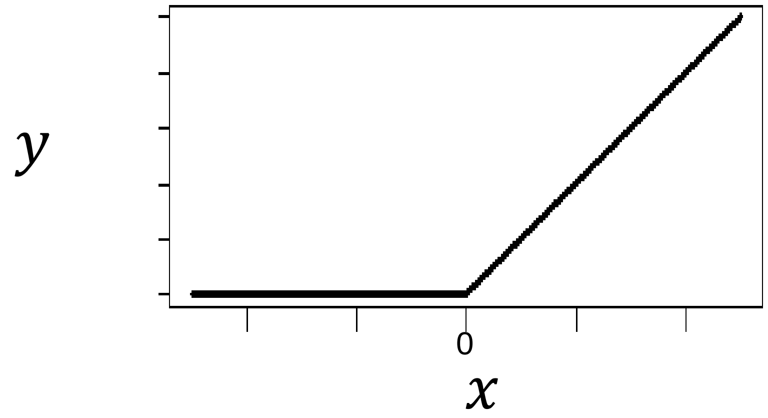




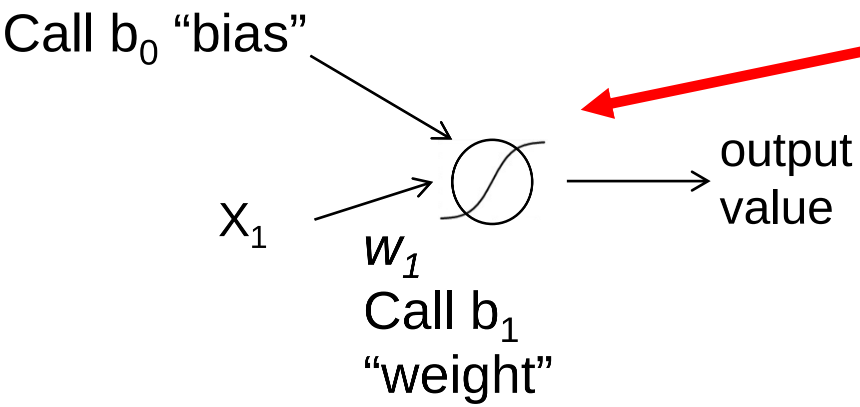
logistic function will transform input to output – call it the 'activation' function

Logistic to Neural Network model

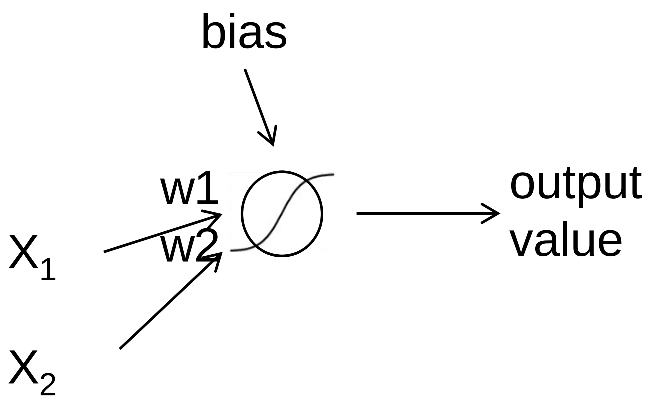
- **RELU activation function**
  - If $(b_o * 1 + b_1 * x) < 0$ set to 0
  - ***And use graphical depiction:***



RELU (rectified linear unit)



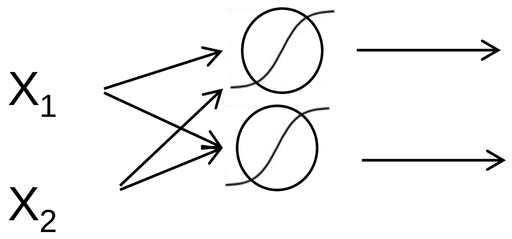
Next step: More general networks



Add input variables

More general networks

(assume bias present)

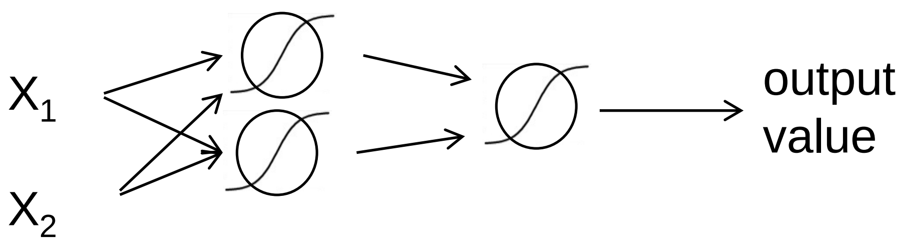


Add input variables

Add logistic transformations …

More general networks

(assume bias)



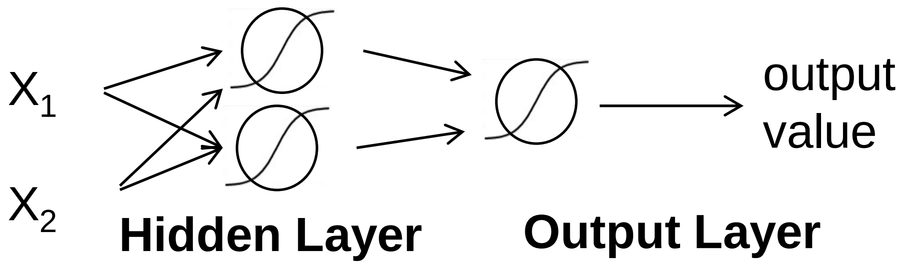
Combine transformations!

Add input variables

Add logistic transformations …

More general networks

(assume bias)

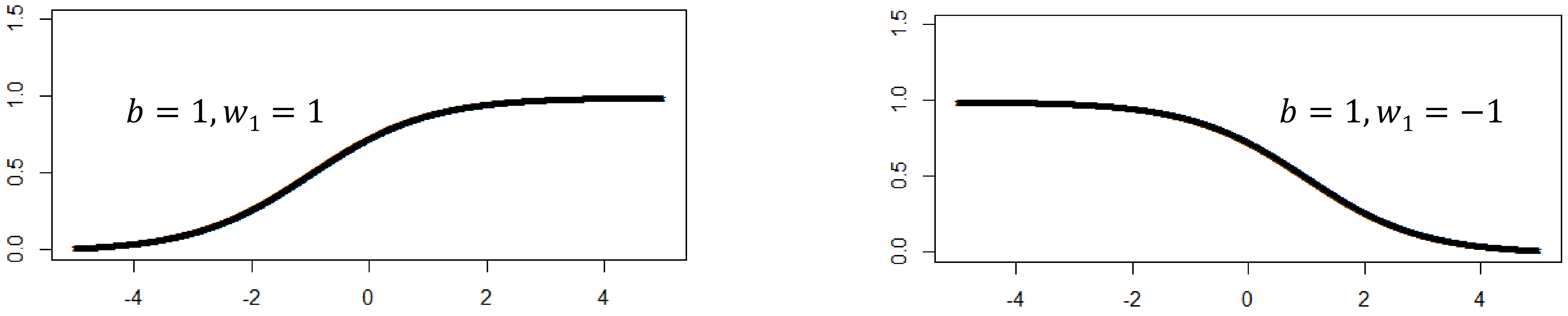


Combine transformations!

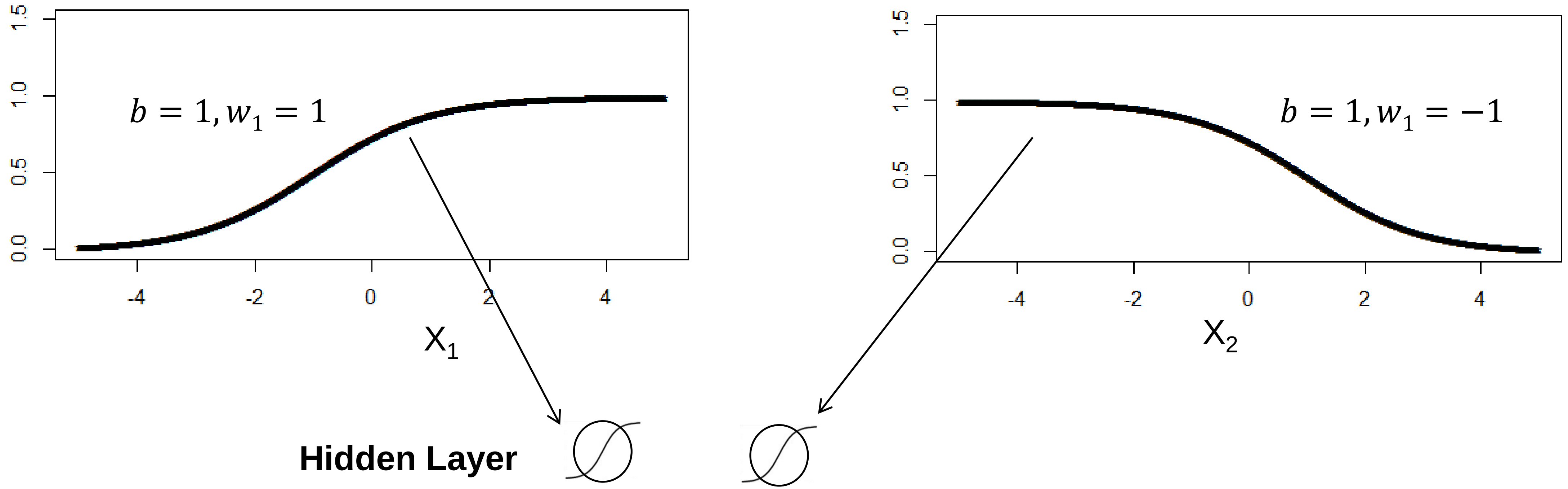
Add input variables

Add logistic transformations …

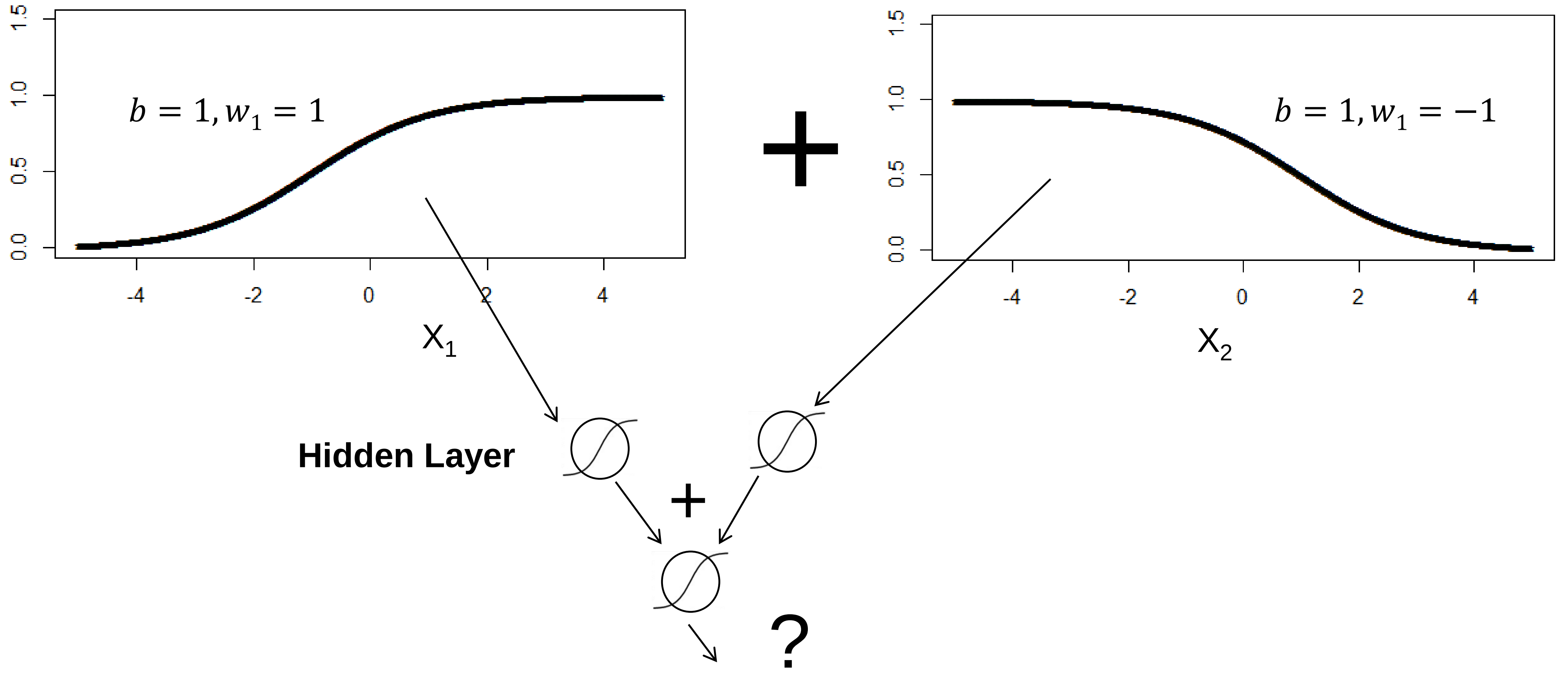
So combinations are highly flexible and nonlinear



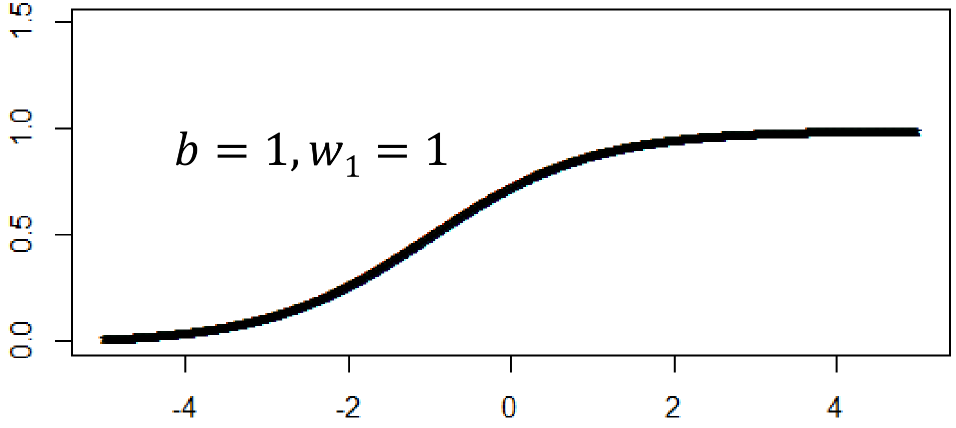
So combinations are highly flexible and nonlinear



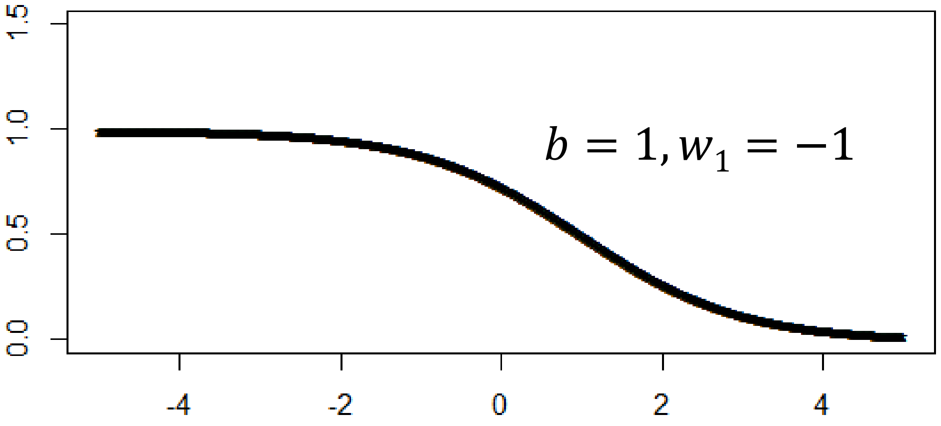
So combinations are highly flexible and nonlinear



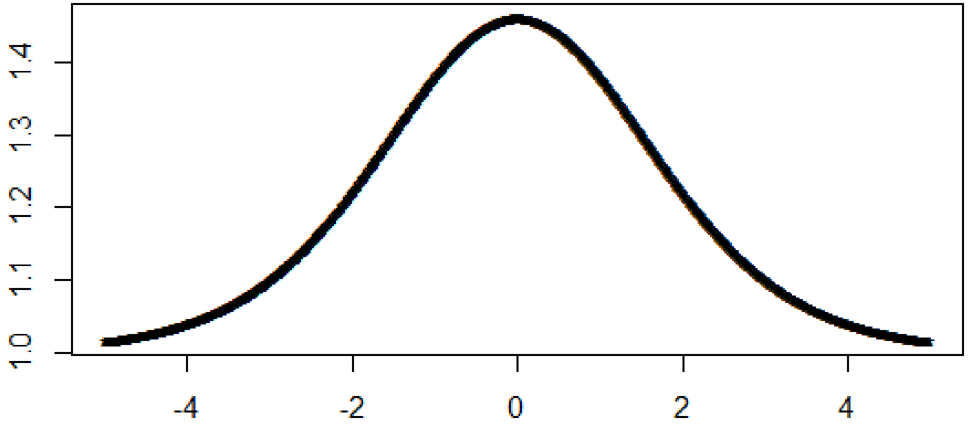
So combinations are highly flexible and nonlinear



+

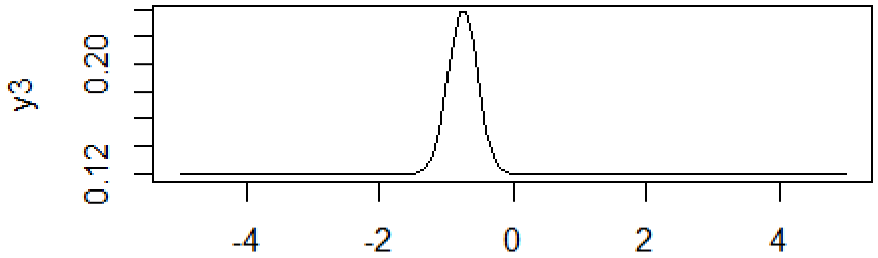


=

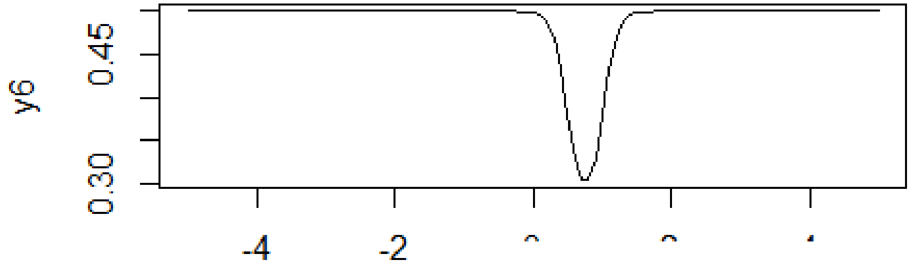


Higher level function combinations

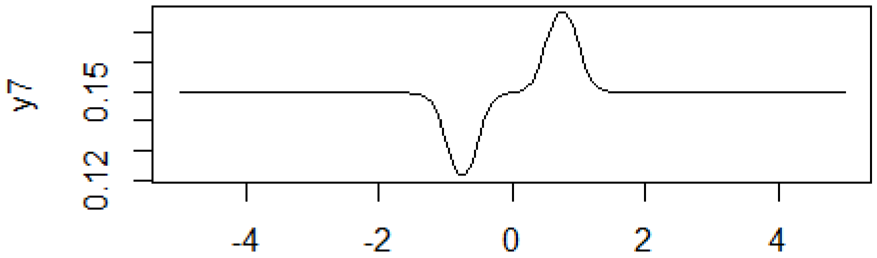
```
x=seq(-5,5,.1)
y1=1/(1+exp(10+   10*x))
y2=1/(1+exp(-5+(-10)*x))
y3=1/(1+exp(1+1*y1+1*y2))
plot(x,y3,type="l")
```




```
y4=1/(1+exp(10+ (-10)*x))
y5=1/(1+exp(-5+(10)*x))
y6=1/(1+exp(1-1*y4-1*y5))
plot(x,y6,type="l")
```

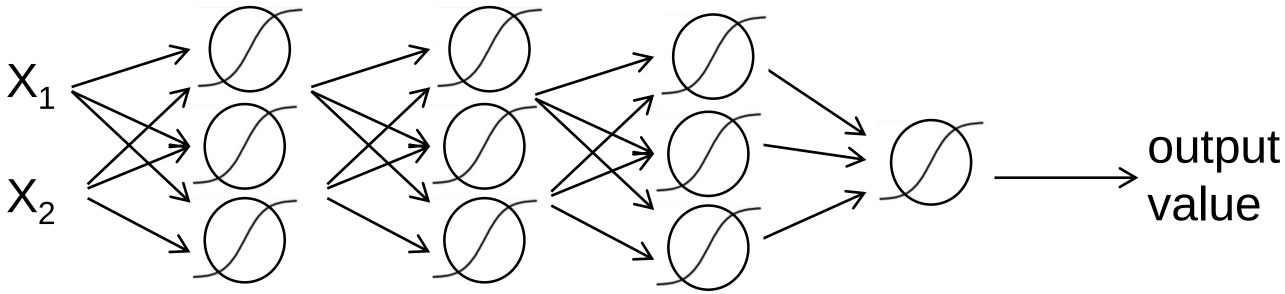



```
y7=1/(1+exp(1+2*y3+1*y6))
plot(x,y7,type="l")
```

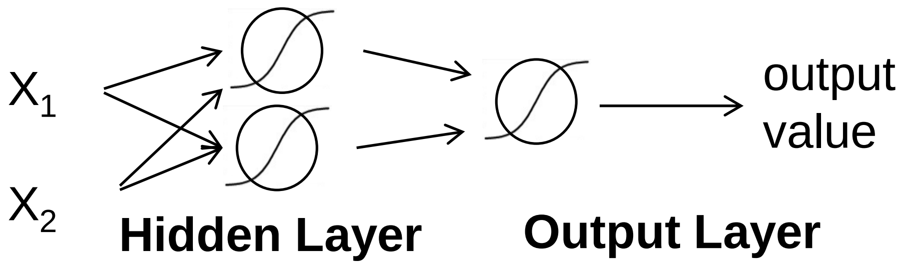


Why stop at 1 hidden layer?

More hidden layers => More varied features and transformation



But parameter fitting is harder too



For each training instance:

Error=Output – Target

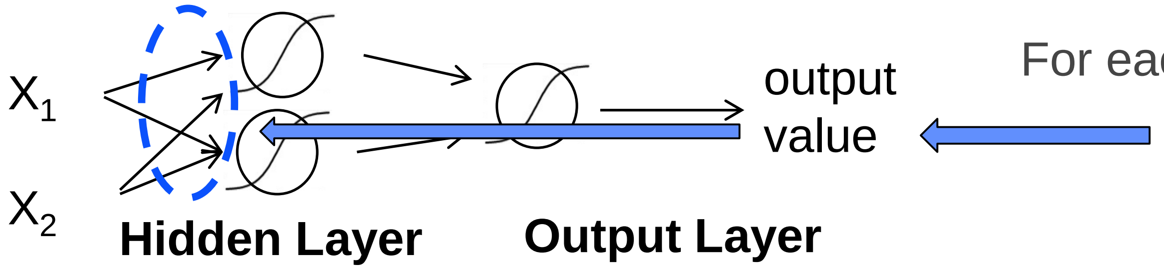
Calculate a *Loss* function of the *Error*

cross-entropy for binary classification

soft-max for multi-classification

root MSE for regression

But parameter fitting is harder too

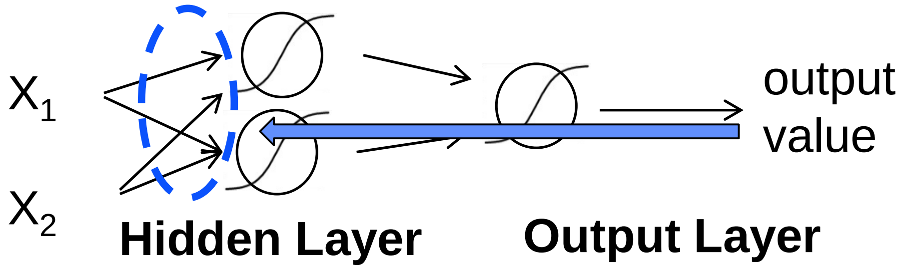


For each training instance:

Error=Output – Target

use derivative chains to ‘back-propagate’ errors

But parameter fitting is harder too



For each training instance:

Error=Output – Target

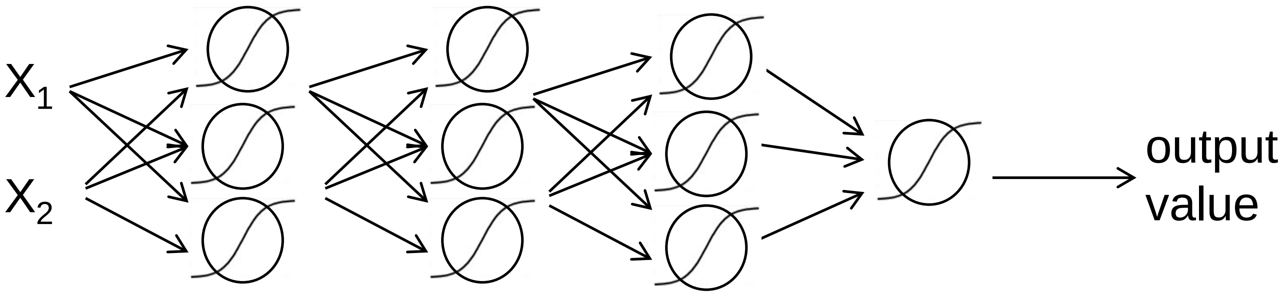
Also, take batches and iterate over whole training set

The method is called:
Stochastic Gradient Descent (sgd)

use derivative chains to 'back-propagate' errors

Train with Care

More hidden layers => More varied features and transformations

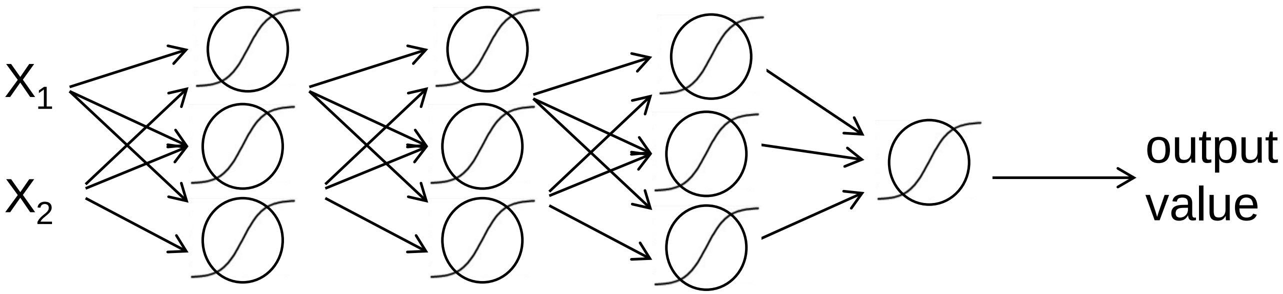


But:

More layers => more parameters

Train with Care

More hidden layers => More varied features and transformations



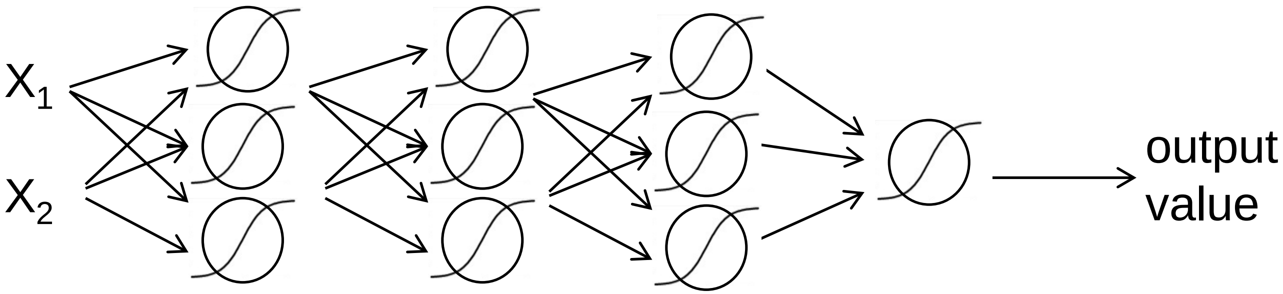
But:

More layers => more parameters => Smaller error for each

***especially* at lower layers**

Train with Care

More hidden layers => More varied features and transformations



But:

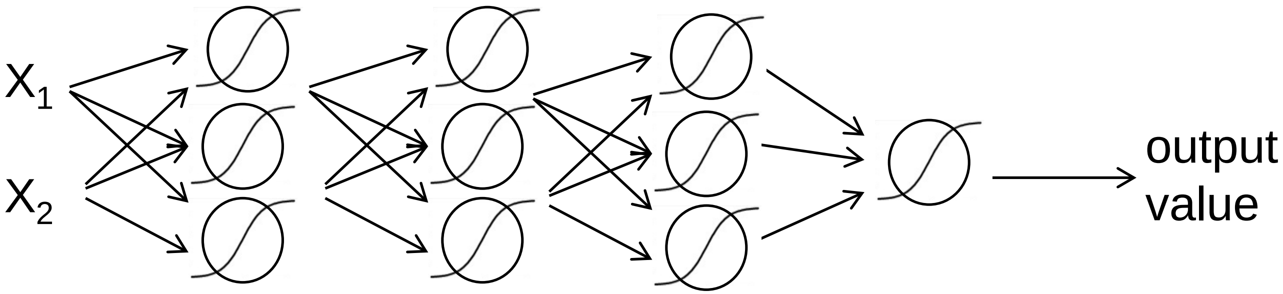
More layers => more parameters => Smaller error for each
***especially* at lower layers**

Need:

More data and computing power (gpu)

Train with Care

More hidden layers => More varied features and transformations



But:

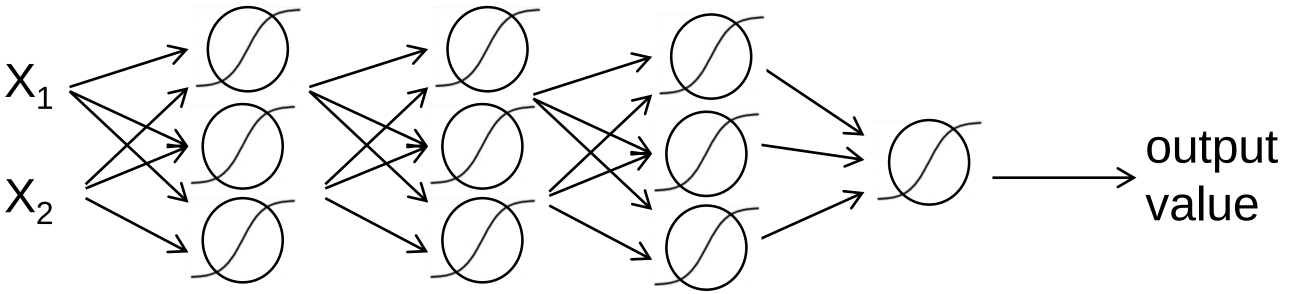
More layers => more parameters => Smaller error for each
especially at lower layers

Need:

More data and computing power (gpu), functions that don't saturate(RELU)

Train with Care

More hidden layers => More varied features and transformations



But:

More layers => more parameters => Smaller error for each
***especially* at lower layers**

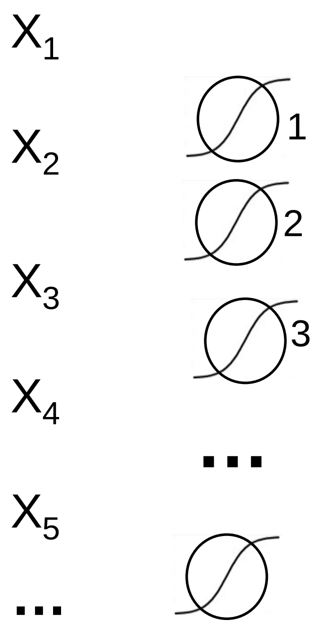
Need:

More data and computing power (gpu), functions that don't saturate(RELU), and ways to avoid over fitting (random node "dropout" or weight penalties)

Feature Transformations, Projections, and Convolutions

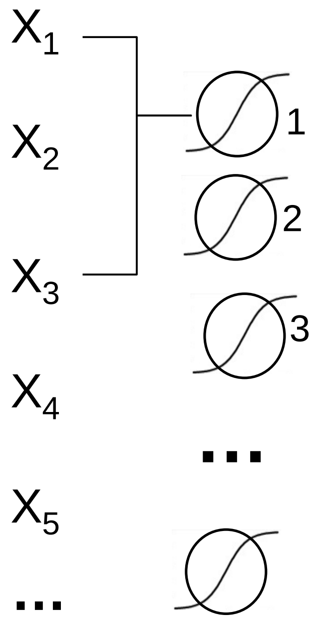
A Filter

Many X input, many hidden nodes, …



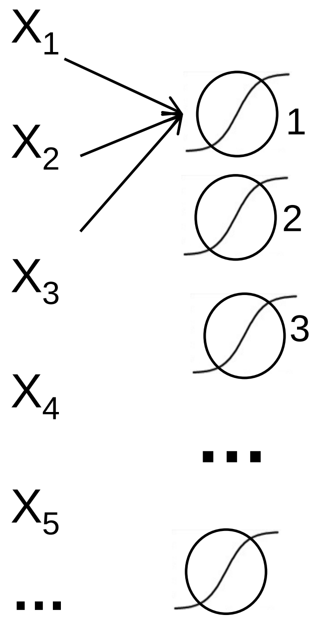
A Filter

Many X input, many hidden nodes, but only local connectivity:



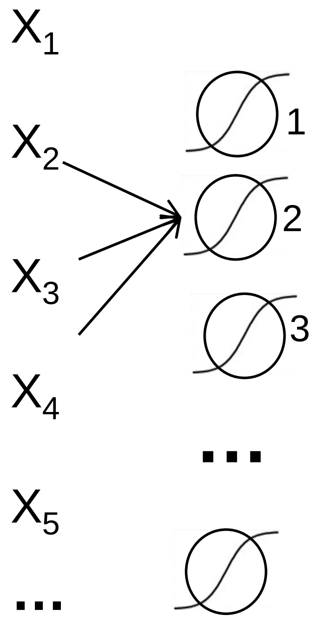
A Filter

Many X input, but only 3 connections to each hidden node
(a local connectivity pattern, aka receptive field)



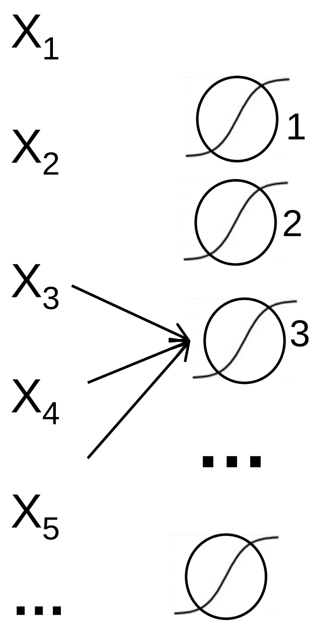
A Filter

Many X input, but only 3 connections to each hidden node
(a local connectivity pattern, aka receptive field)



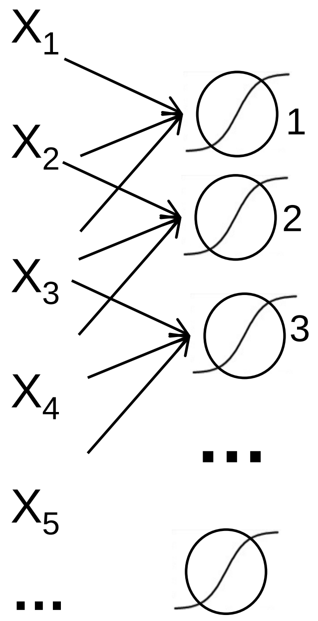
A Filter

Many X input, but only 3 connections to each hidden node
(a local connectivity pattern, aka receptive field)



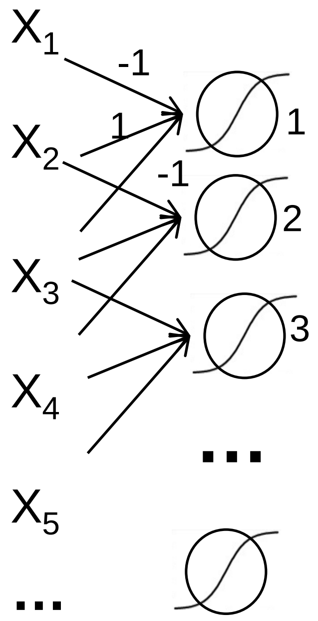
A Filter

Many X input, but only 3 connections to each hidden node
(a local connectivity pattern, aka receptive field)



A Filter

Many X input, but only 3 connections to each hidden node
(a local connectivity pattern, aka receptive field)

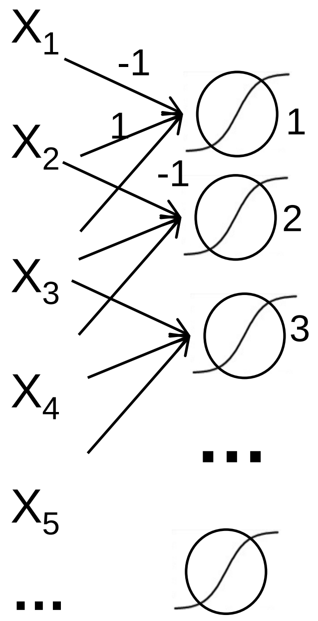


For node 1 let W= [w_1 w_2 w_3] = [-1 1 -1]

What is the node 1 doing?

A Filter

Many X input, but only 3 connections to each hidden node
(a local connectivity pattern, aka receptive field)



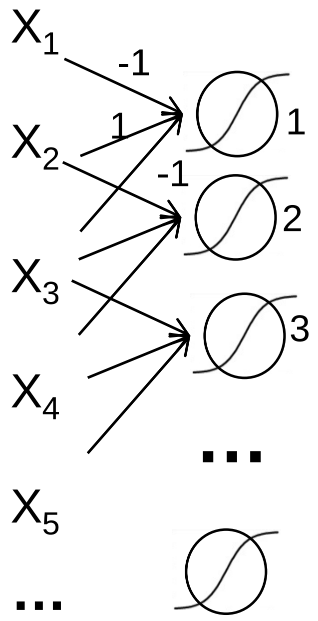
For node 1 let W= $[w_1\, w_2\, w_3\,] = [-1\ 1\ -1]$

What is the node 1 doing?

Informally, node 1 has max activation for a ‘spike’, e.g. when X_2 is positive and X_1, X_3 are negative

A Filter

Many X input, but only 3 connections to each hidden node
(a local connectivity pattern, aka receptive field)



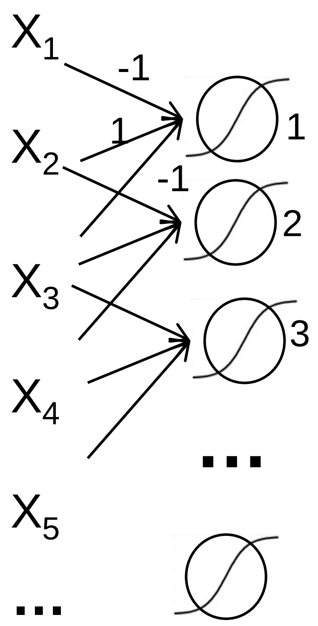
For node 1 let W= [w_1 w_2 w_3] = [-1 1 -1]

For node 2,3, etc… copy W for node 1 so that node 2 and 3 are looking for spikes in their “receptive” field

What is the hidden layer doing?

A Filter

Many X input, but only 3 connections to each hidden node
(a local connectivity pattern, aka receptive field)



For node 1 let W=[w_1 w_2 w_3] = [-1 1 -1]

For node 2,3, etc… copy W for node 1 so that node 2 and 3 are looking for spikes in their “receptive” field

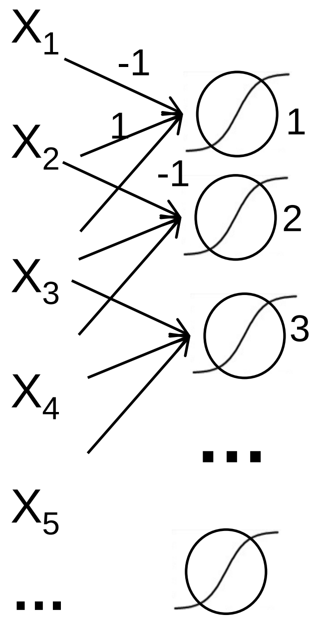
What is the hidden layer doing?

Informally, looking for a spike everywhere.

This is essentially a convolution operator, where W is the kernel.

A Filter

Many X input, but only 3 connections to each hidden node
(a local connectivity pattern, aka receptive field)



For node 1 let W=[w_1 w_2 w_3] = [-1 1 -1]

For node 2,3, etc… copy W for node 1 so that node 2 and 3 are looking for spikes in their “receptive” field

What is the hidden layer doing?

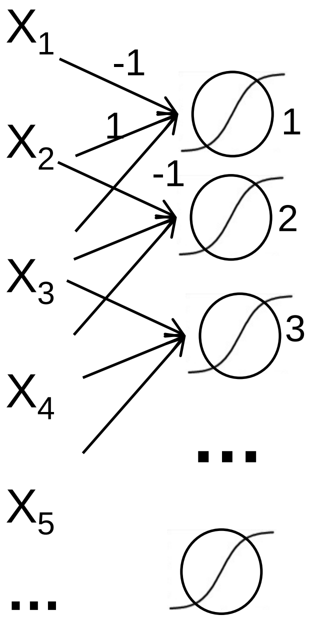
Informally, looking for a spike everywhere.

This is essentially a convolution operator, where W is the kernel.

Note: copying weights is like *sliding* W across input

A Filter

Many X input, but only 3 connections to each hidden node
(a local connectivity pattern, aka receptive field)



For node 1 let W=[w_1 w_2 w_3] = [-1 1 -1]

For node 2,3, etc… copy W for node 1 so that node 2 and 3 are looking for spikes in their “receptive” field

What is the hidden layer doing?

Informally, looking for a spike everywhere.

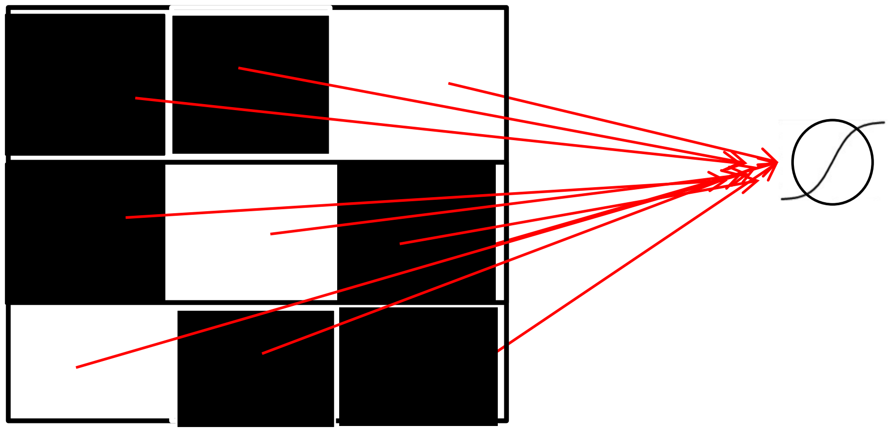
This is essentially a convolution operator, where W is the kernel

Note: copying weights is like *sliding* W across input

Note: if we take max activation across nodes (‘Max Pool’) then it’s like looking for a spike *anywhere*.

2D Convolution

Now let input be a 2D binary matrix, e.g. a binary image, fully connected to 1 node

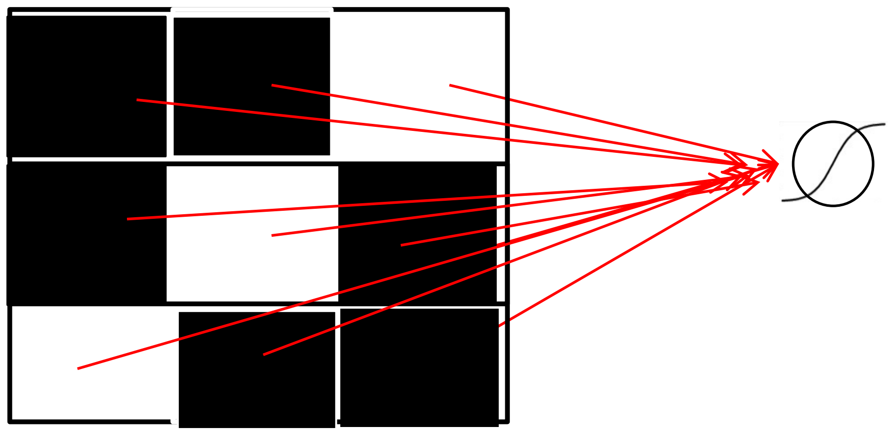


(black= -1 white=1)

What W matrix would 'activate' for a upward-toward-left diagonal line?

2D Convolution

Now let input be a 2D binarized 3x3 matrix fully connected to 1 node



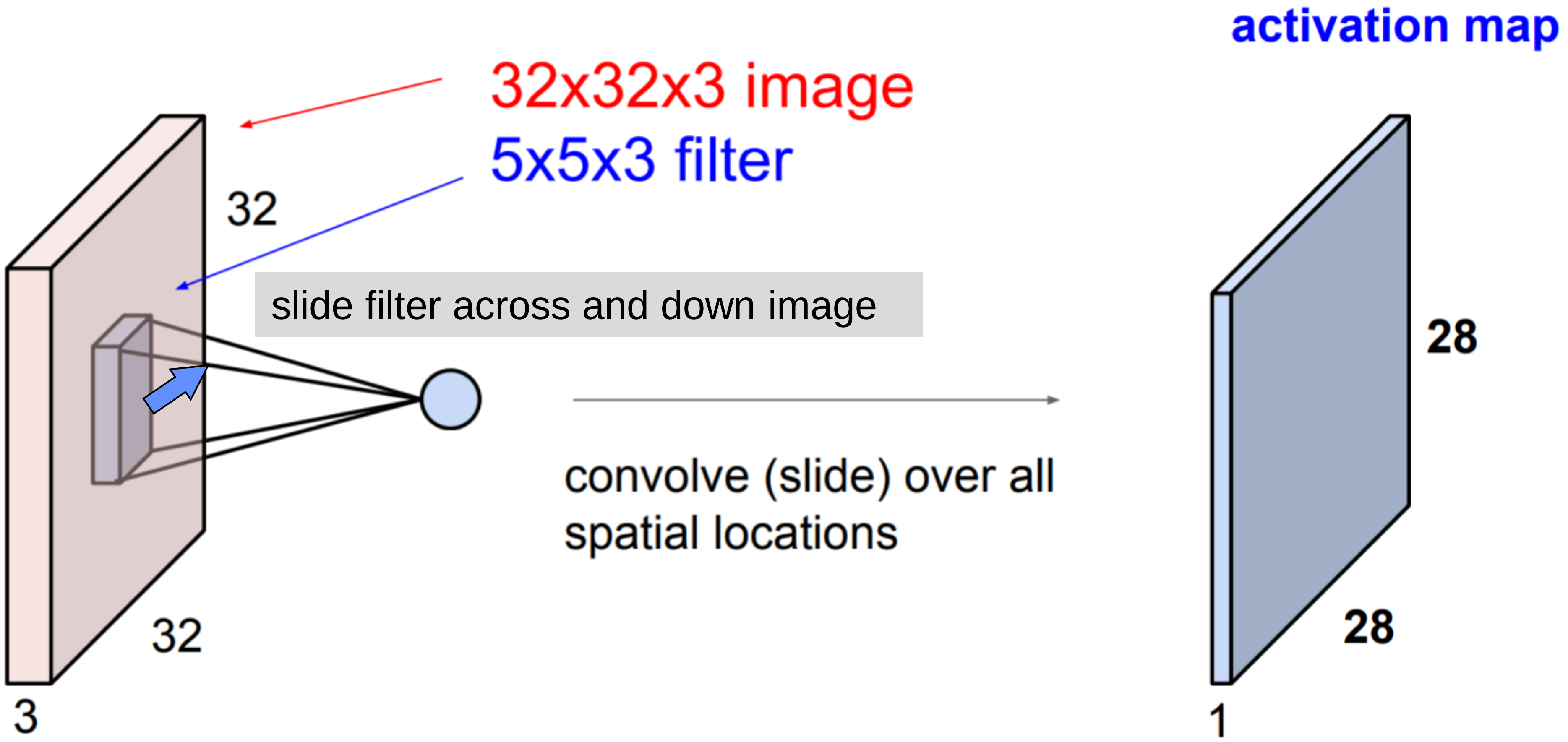
(black= -1 white=1)

What W matrix would 'activate' for a upward-toward-left diagonal line?

How about:

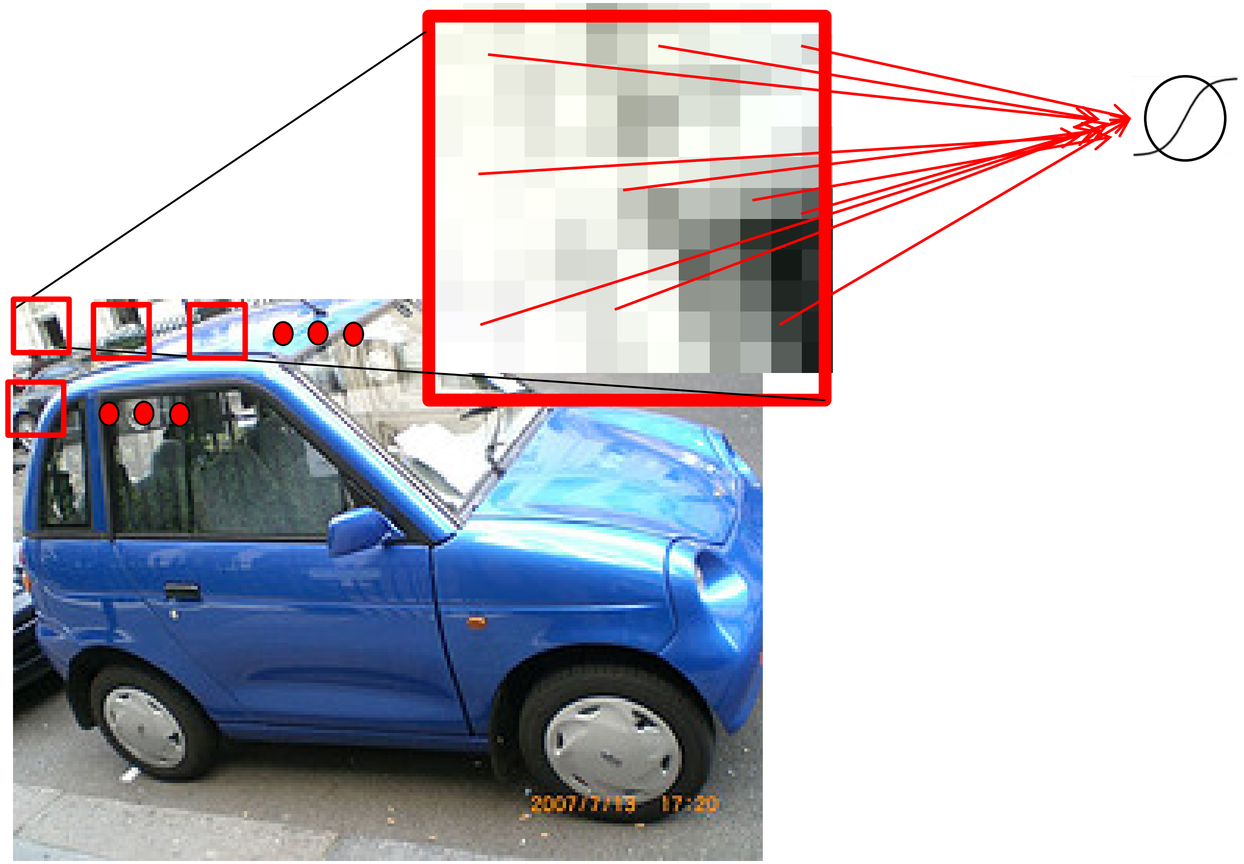
$$W = \begin{bmatrix} -1 & -1 & 1 \\ -1 & 1 & -1 \\ 1 & -1 & -1 \end{bmatrix}$$

2D Convolution of Image to Feature map



2D Convolution

For full image, 1 filter is applied to 1 region in 1 color channel at a time, and then slid across regions (or done in parallel with shared weights) and produces 1 new 2D image (hidden) layer



Convolution Layer parameters:

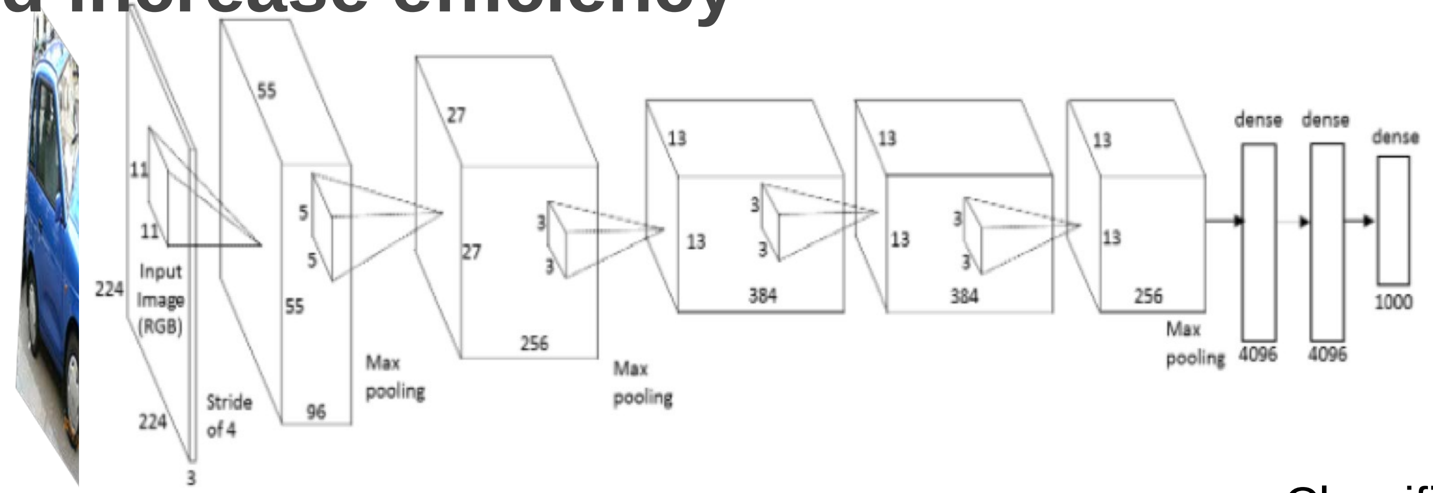
- filter size depends on input:
  - smaller filters for smaller details
  - 2 layers of 3x3 ~ 1 layer of 5x5
- sliding amount
  - smaller better but less efficient
- number of filters
  - depends on task
  - each filter is a new 2D layer

Convolution Network :

many layers and architecture options

Large Scale Versions

- Large (deep) Convolution Networks are turning out to be feasible with GPUs (some are 100+ layers)
- Need large amounts of data and many heuristics to avoid overfitting and increase efficiency

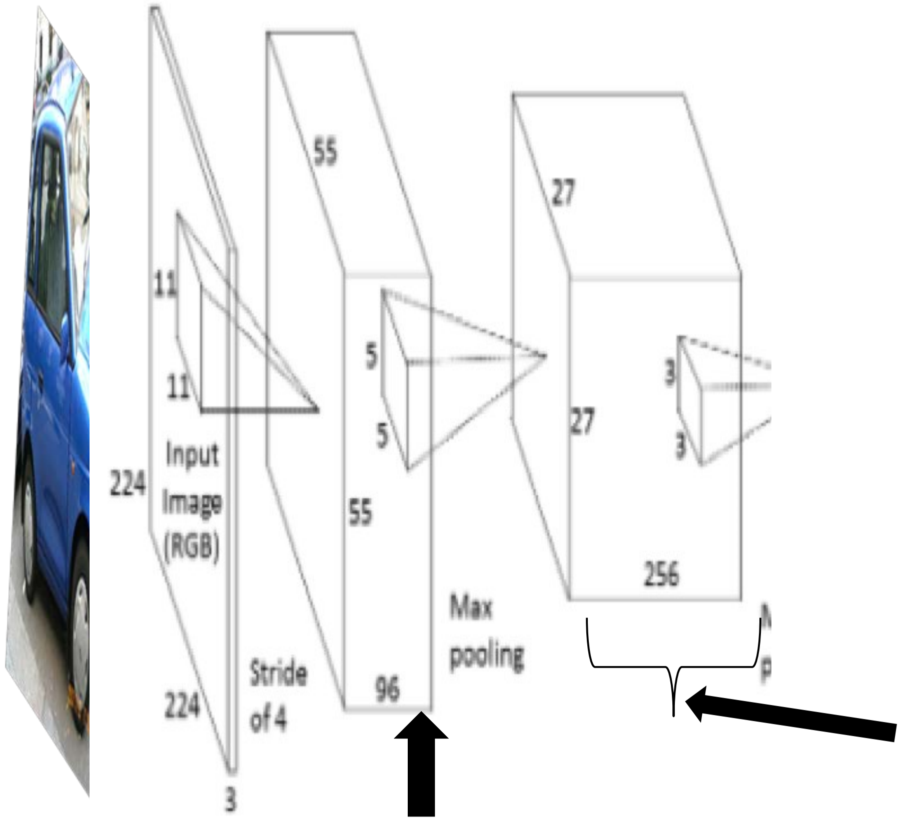


Convolution layers

Classification layers and output

Large Scale Versions

- **Zooming in: Convolution layers**



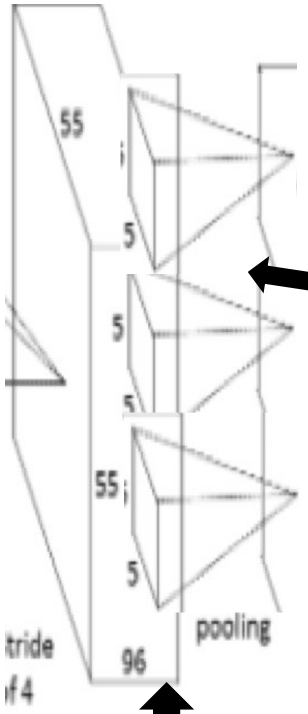
The thickness is the number of different convolutions, i.e. different transformations, sometimes called "channels"

Each convolution layer uses RELU (rectified linear activation units instead of logistic function) and is followed by Max Pooling layer (over 2D regions with sliding)

Large Scale Versions

- **Zooming in:**

Max pooling



Max pooling: take the maximum over a region and slide the region.

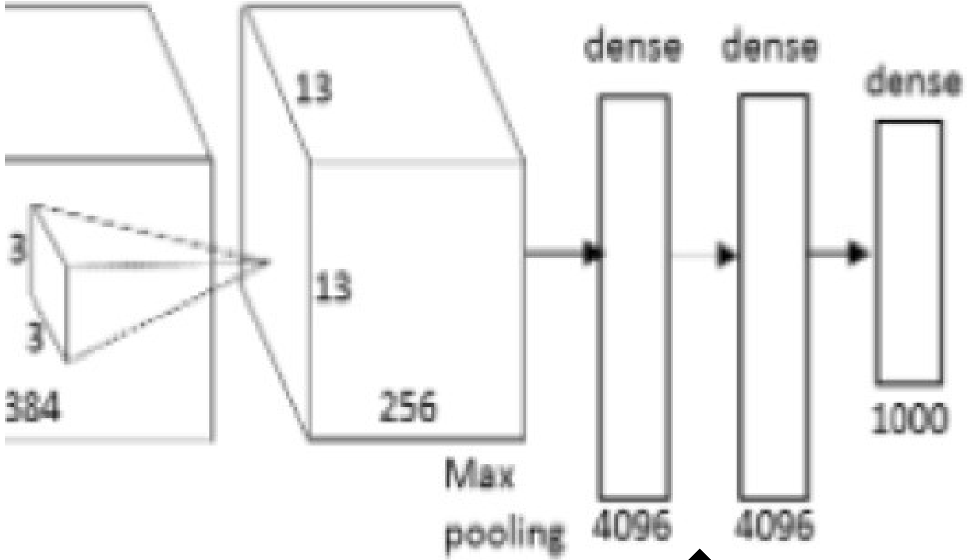
The larger the slide, the more down sampling occurs (which helps compute time)

Each convolution layer uses RELU (rectified linear activation units instead of logistic function) and is followed by Max Pooling layer (over 2D regions with sliding)

Large Scale Versions

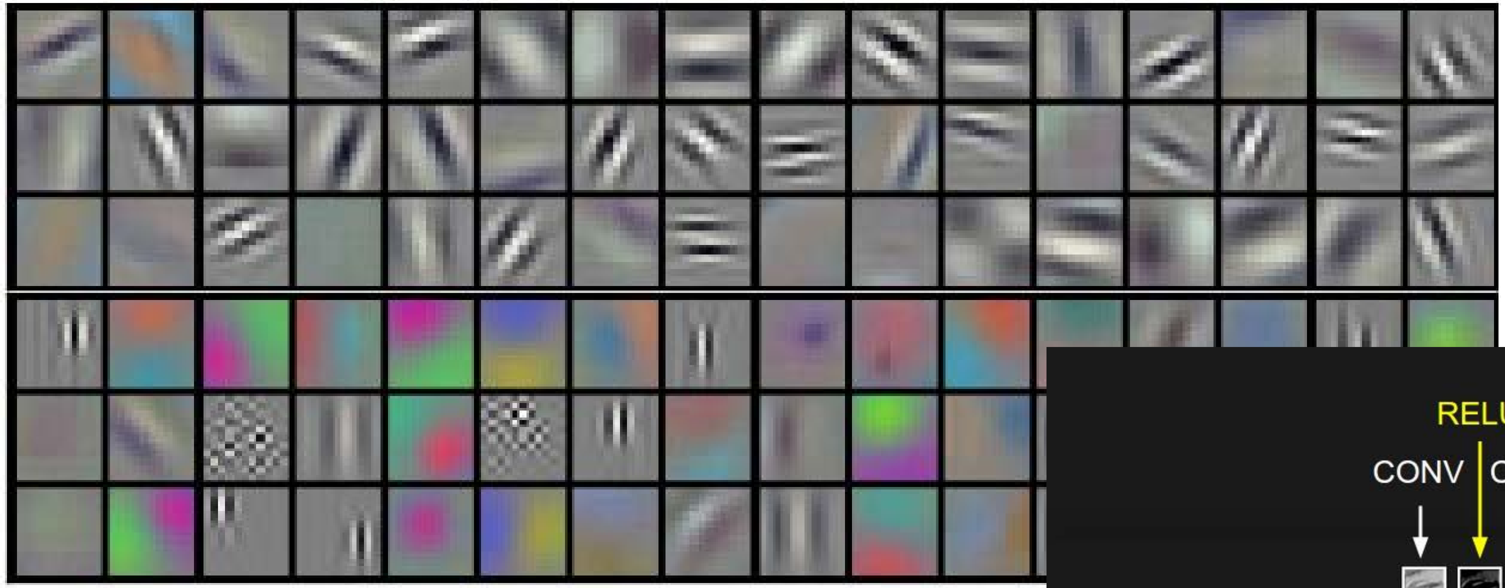
- **Zooming in:**

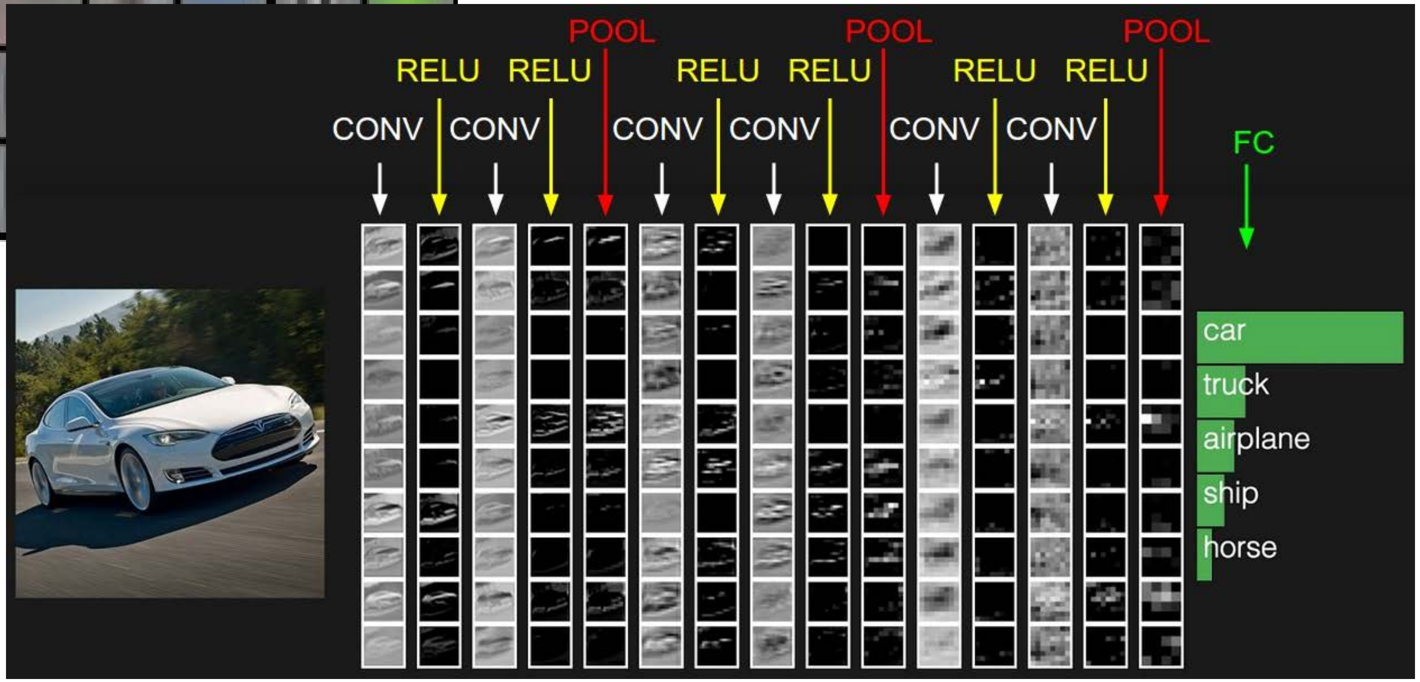
Classification layers



Last convolution layer is laid out as a vector for input into classification layers.
Classification uses dense, i.e. fully connected, hidden layers and output layer.

What Learned Convolutions Look Like





Summarizing Deep Layers

- **Hidden layers transform input into new features:**
  - Feature can be highly nonlinear
  - Features as a new space of input data
  - Features as projection onto lower dimensions (compression)
  - Features as filters, which can be used for convolution
- **But also:**
  - Many algorithm parameters
  - Many weight parameters
  - Many options for stacking layers

Feature Coding vs Discovery

- Edge detection with Support Vector Machine

  OR

  Convolution Neural Network?

- With small datasets and reasonable features, SVMs can work well

- Large classification problems can benefit from common features that CNNs can discover

Pause

What is Transfer Learning?

- **To overcome challenges of training model from scratch:**
  - Insufficient data
  - Very long training time
- **Use pre-trained model**
  - Trained on another dataset
  - This serves as starting point for model
  - Then train model on current dataset for current task

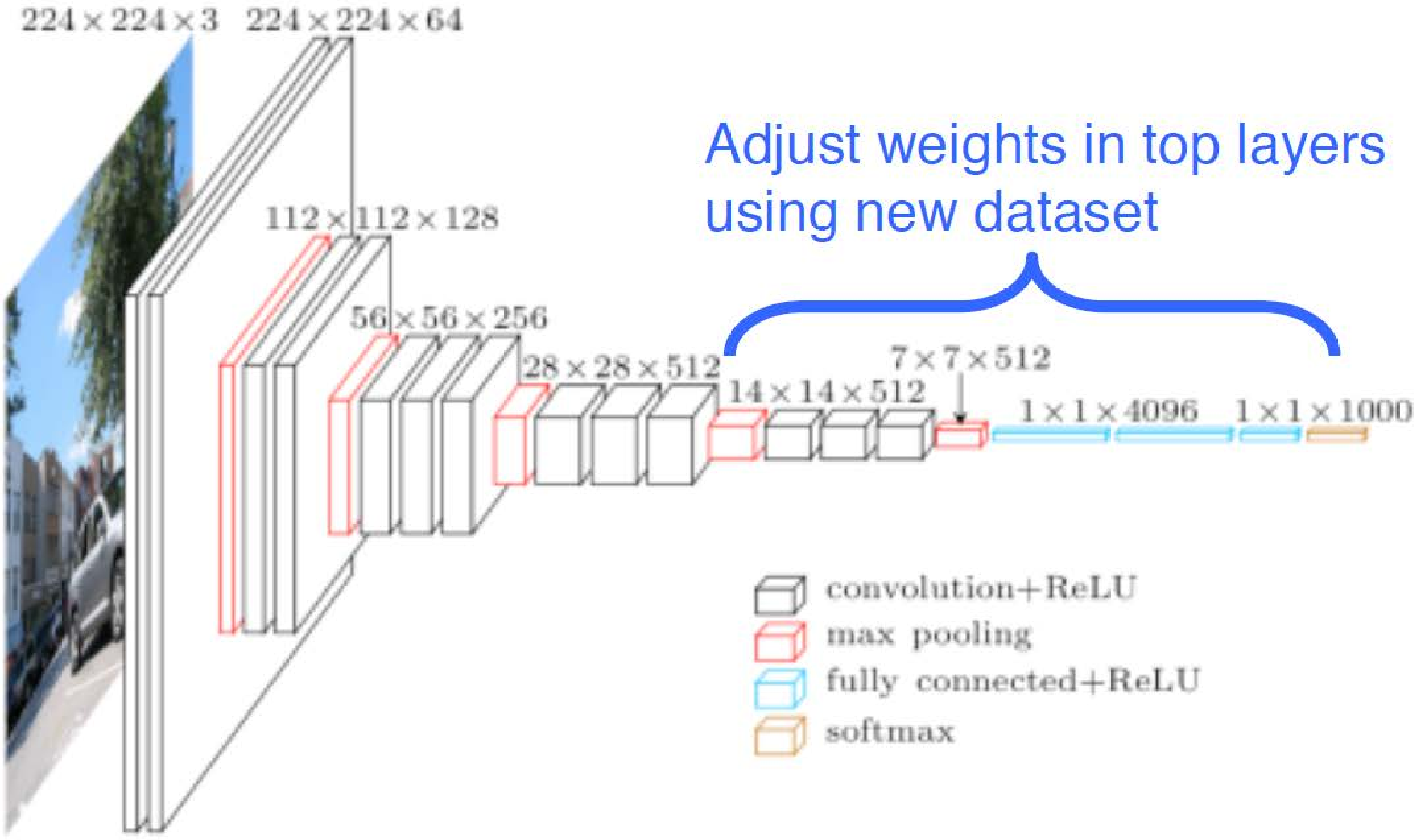
Transfer Learning Approaches

- **Feature extraction**
  - Remove last fully connected layer from pre-trained model
  - Treat rest of network as feature extractor
  - Use features to train new classifier (“top model”)
- **Fine tuning**
  - Tune weights in some layers of original model (along with weights of top model)
  - Train model for current task using new dataset

CNNs for Transfer Learning

- **Popular architectures**
  - AlexNet
  - GoogLeNet
  - VGGNet
  - ResNet
- **All winners of ILSVRC**
  - ImageNet Large Scale Visual Recognition Challenge
  - Annual competition on vision tasks on ImageNet data

Transfer Learning – Fine Tuning



Works best when new data is similar to original data, else use lower layers and more retraining.

Source: https://www.cs.toronto.edu/~frossard/post/vgg16/

The Zoo

- Machine learning/convolution network frameworks:

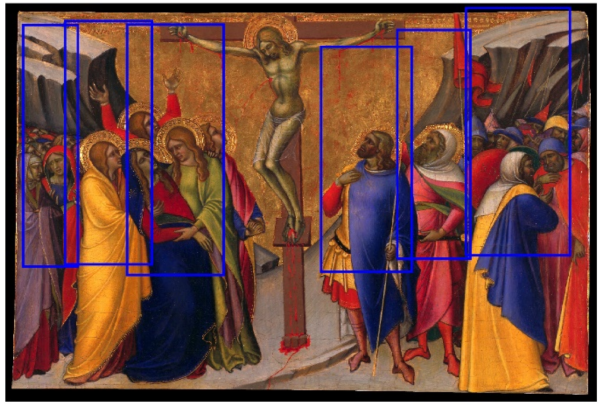
  Tensorflow, pyTorch (libraries and API to build graphs of networks and processing)

  Keras - higher level CNN library with tensorflow (best for learning)
  Caffe – C/C++ library with many pretrained models
  Caffe2 – Facebook tookover Caffe, Pytorch (has a good model for people detection)
  YOLO/Darknet – A C++ library, with object detection
  Matlab – CNN functions, and pretrained networks

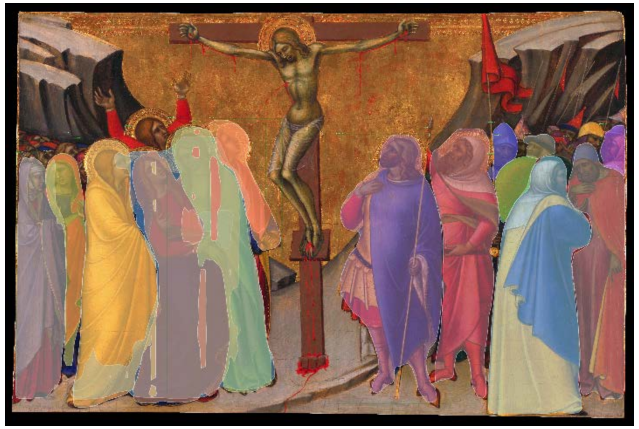
- Many networks pretrained on large or particular object classes are available: AlexNet, VGG19, Googlenet, Detectron
- Training CNNs require GPUs; CPUs are fine running pretrained CNNs
- Big Tech have online services (see next page)

Caffe2, Facebook “Detectron” networks

Object Detection
ie getting a region
bounding box
(rcnn)



Object
Segmentation
ie getting a mask
(mask-rcnn)



Object Parts
ie getting keypoints
(keypoint-rcnn)

Caffe2 Detectron on Comet

- **git clone *https://github.com/facebookresearch/Detectron***

  You will get folders of tools, utilities, etc..

- **On Comet compute node, run:**

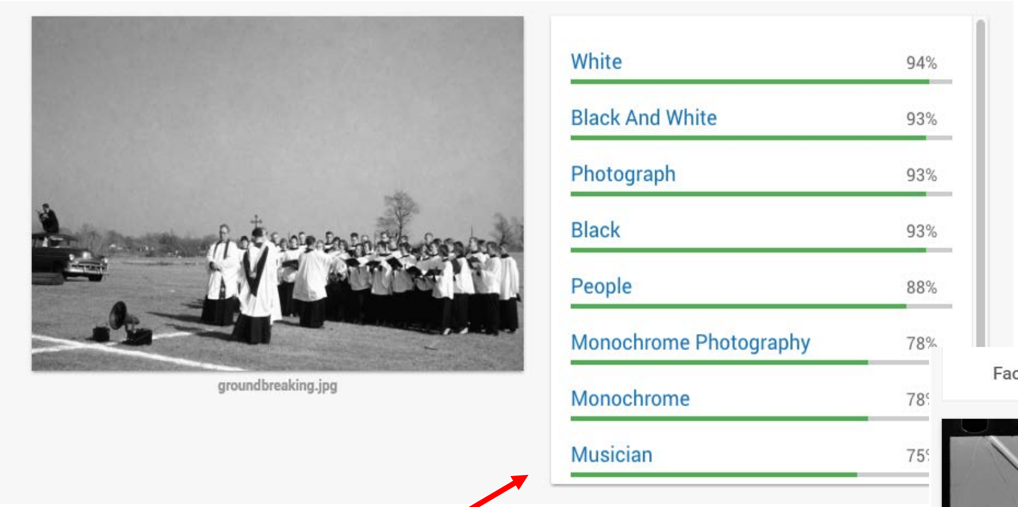
  module purge

  module load singularity

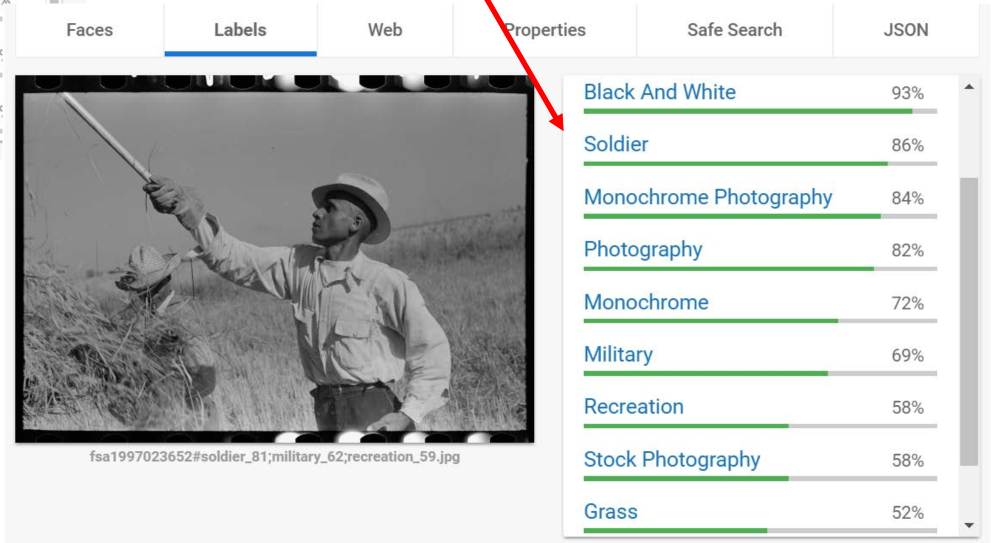
  singularity shell /share/apps/gpu/singularity/images/pytorch/pytorch-v1.0.0-gpu-20190110.simg

Google tool for objects, faces, text

- Google Vision api – object recognition network



fails on “Soldier”



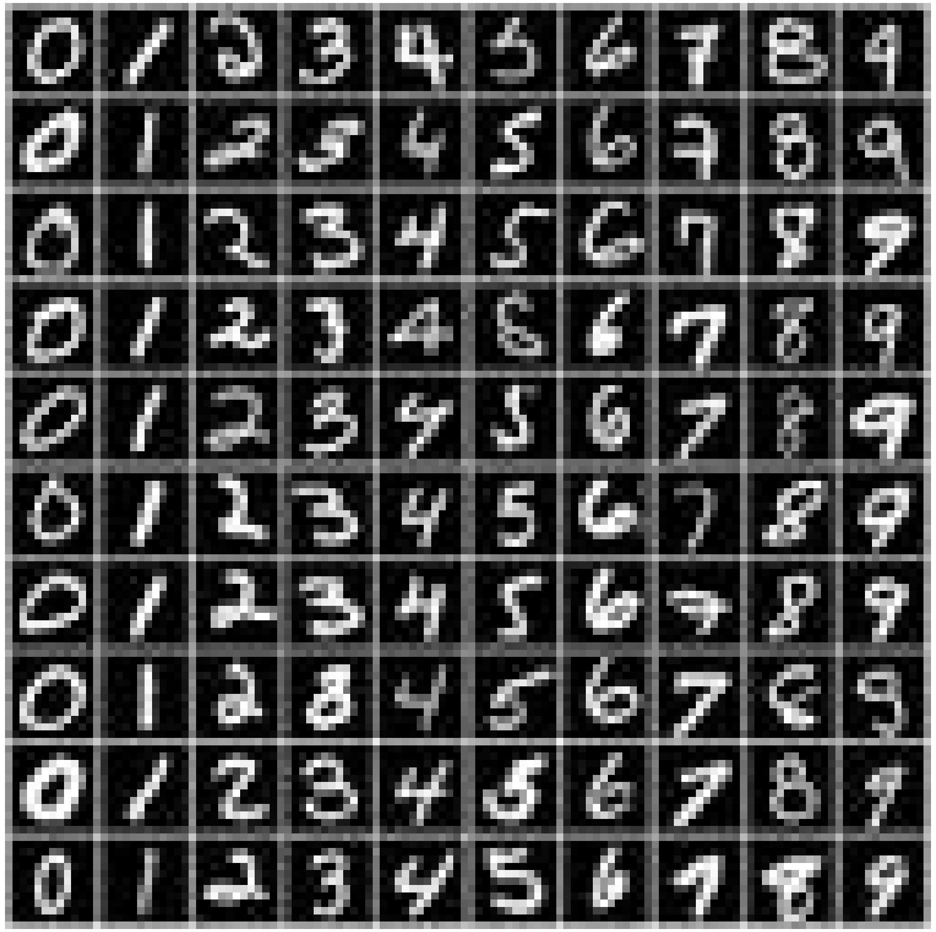
gets “Musician”

References

- **Book: https://mitpress.mit.edu/books/deep-learning**
- **F. Chollet "Deep Learning with Python"**
- **Documentation: https://keras.io/**
- **Tutorials I used (borrowed):**
  - http://cs231n.github.io/convolutional-networks/
  - https://hackernoon.com/visualizing-parts-of-convolutional-neural-networks-using-keras-and-cats-5cc01b214e59
  - https://github.com/julienr/ipynb_playground/blob/master/keras/convmnist/keras_cnn_mnist.ipynb

Tutorial

- **MNIST database of handwritten printed digits**
- **The ‘hello world’ of Conv. Neural Networks**
- **Use Keras front end (high level neural functions) to Tensorflow engine (neural math operations)**
- **Works with GPU or CPUs**



MNIST on Comet

- **Login and get an interactive compute node session**

- **Start up conda python environment**

```
. /share/apps/compute/si2019/miniconda3/etc/profile.d/conda.sh
```

(^ yes, it's a 'dot' followed by space)

```
conda activate
jupyter notebook --no-browser --ip="*" &
```

Cut and paste http address, edit localhost, look in DeepLearningTutorial for notebook

Home | LabMNIST_Final

comet-18-14.sdsc.edu:8888/notebooks/LabMNIST_Final.ipynb

jupyter LabMNIST_Final Last Checkpoint: 7 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Help

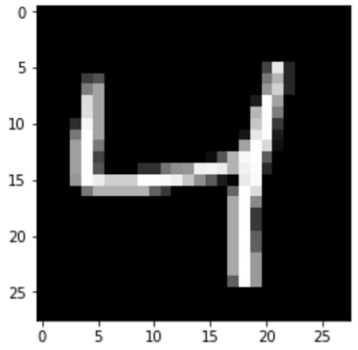
Not Trusted

Python 3

Code

```python
for i in range(0,3):
    im = Image.fromarray(X_train[i,:,:])
    im.save("Xtrain_num"+str(i)+"_cat_"+str(Y_train[i])+".jpeg")

plt.figure()
plt.imshow(im,'gray')
plt.show()

print('img load done')
print (time.strftime("%H:%M:%S"))
```

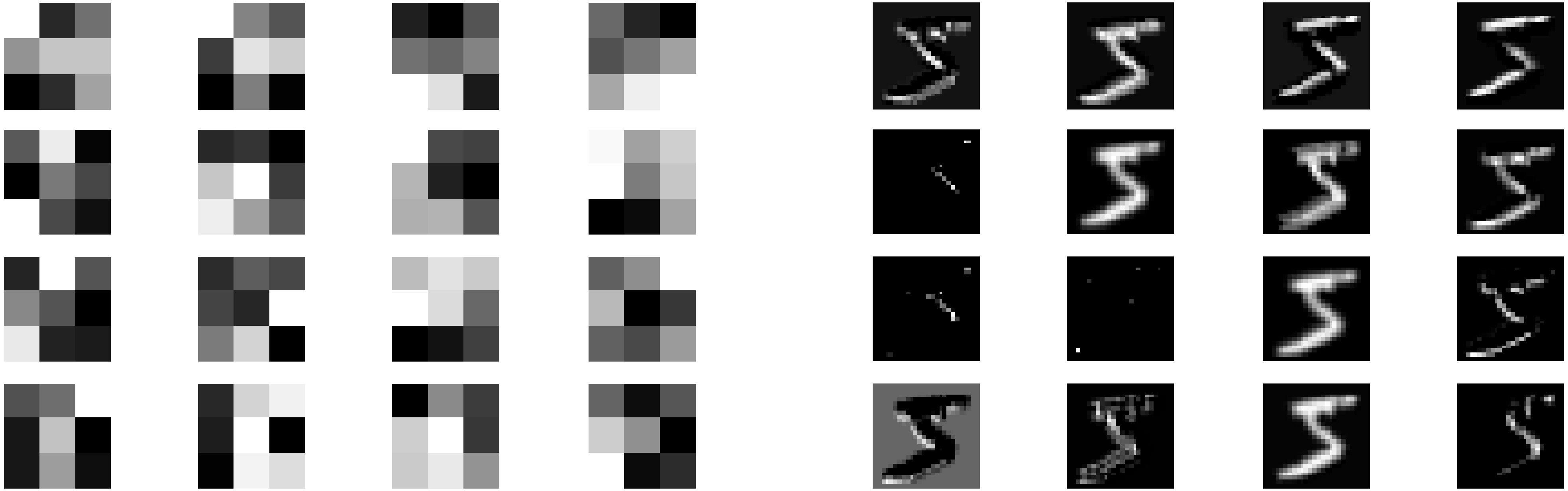
```
(5000, 28, 28)

<matplotlib.figure.Figure at 0x2ad45d04f2e8>
```



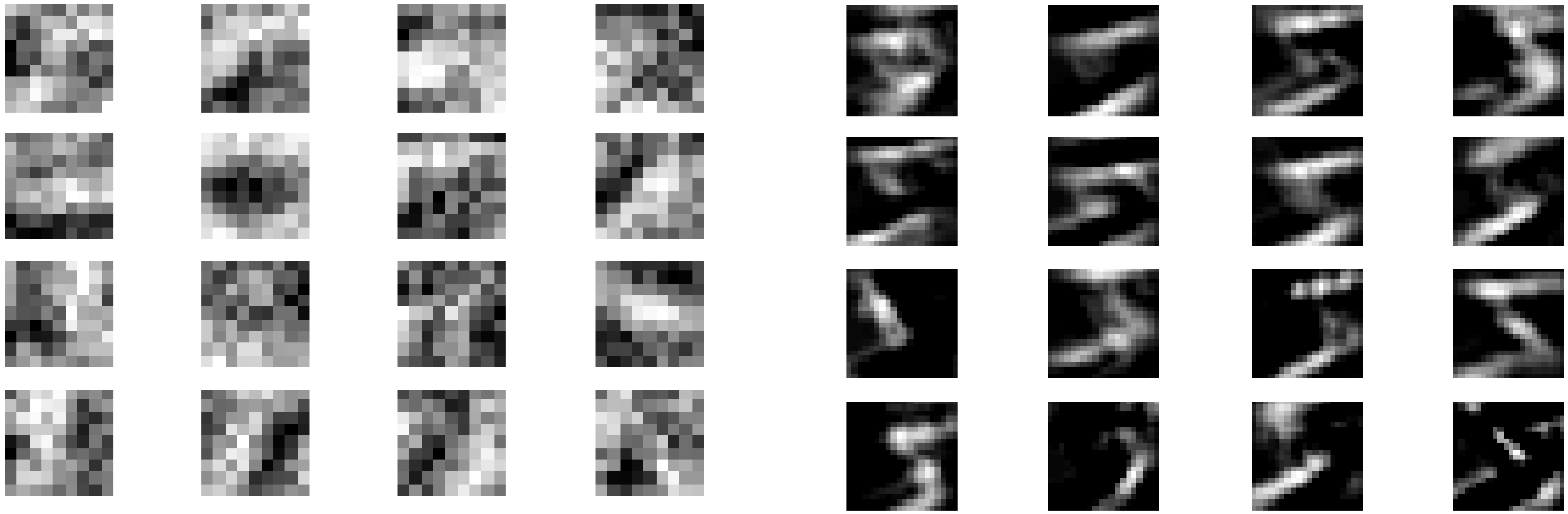

```
img load done
```

10:09 PM
7/31/2017

3x3 first convolution layer filter and activation



9x9 first convolution layer filter and activation



Pause