Knet.jl

Beginning Deep Learning with 100 lines of Julia

Deniz Yuret

denizyuret@gmail.com
http://www.denizyuret.com
@denizyuret

Outline

- Why Knet/Julia for deep learning?
- Machine learning in 5 slides
- 5 example models
- KnetArray: yet another GPU array type
- AutoGrad: how do we differentiate all of Julia?

Grzegorz Chrupała
@gchrupala

Following

Have to wean myself off of #Theano. What should I switch to? #Pytorch #Dynet

69% Pytorch

31% Dynet

95 votes • Final results

Retweet 1 Likes 6

5:02 AM - 8 Jun 2017

10 1 6

Deniz Yuret
@denizyuret

Replying to @gchrupala

May I suggest github.com/denizyuret/Kne…? Dynamic graphs, gpu support, convnets (working vgg and resnet examples included).

denizyuret/Knet.jl
Koç University deep learning framework. Contribute to Knet.jl development by creating an account on GitHub.
github.com

Like
1

9:58 AM - 8 Jun 2017

1 1

Grzegorz Chrupała @gchrupala · Jun 8

Replying to @denizyuret

Sounds cool. But Julia?

1

Deniz Yuret @denizyuret · Jun 8

Julia is awesome!

2

Grzegorz Chrupała @gchrupala · Jun 8

Possibly, but students and co workers. Would have to be fucking amazing to switch the whole lab from Python/numpy.

1 1 2

Deniz Yuret @denizyuret · Jun 8

Took me 2 yrs and a grad course. Now students can implement e.g. arxiv.org/abs/1706.01427 in a day. Hope will accelerate research.

1

Why Knet.jl?

Development and Usage

- Pure Julia (with some cuda libraries for gpu).
- Actively developed and used for 2 years.
- Teaching
  - Students replicate arxiv models within a semester!
  - Projects include: image captioning, object recognition, sketch recognition, DNA sequence conservation, multi-lingual parsing, chatbots…
- Research
  - Natural language processing
  - Image processing
  - MLP, RNN, CNN, RL, attention based models...

Performance (single GPU)

model	dataset	epochs	batch	Knet	Theano	Torch	Caffe	TFlow
LinReg	Housing	10K	506	2.84	1.88	2.66	2.35	5.92
Softmax	MNIST	10	100	2.35	1.40	2.88	2.45	5.57
MLP	MNIST	10	100	3.68	2.31	4.03	3.69	6.94
LeNet	MNIST	1	100	3.59	3.03	1.69	3.54	8.77
CharLM	Hiawatha	1	128	2.25	2.42	2.23	1.43	2.86

What is Knet.jl?

What is Knet.jl?

Julia + little-else!

What is Knet.jl?

Julia + little-else!

Write models in Julia
rather than a weak mini-language

What is Knet.jl?

Julia + little-else!

Write models in Julia
rather than a weak mini-language

Little else =
(some gpu support & autograd)

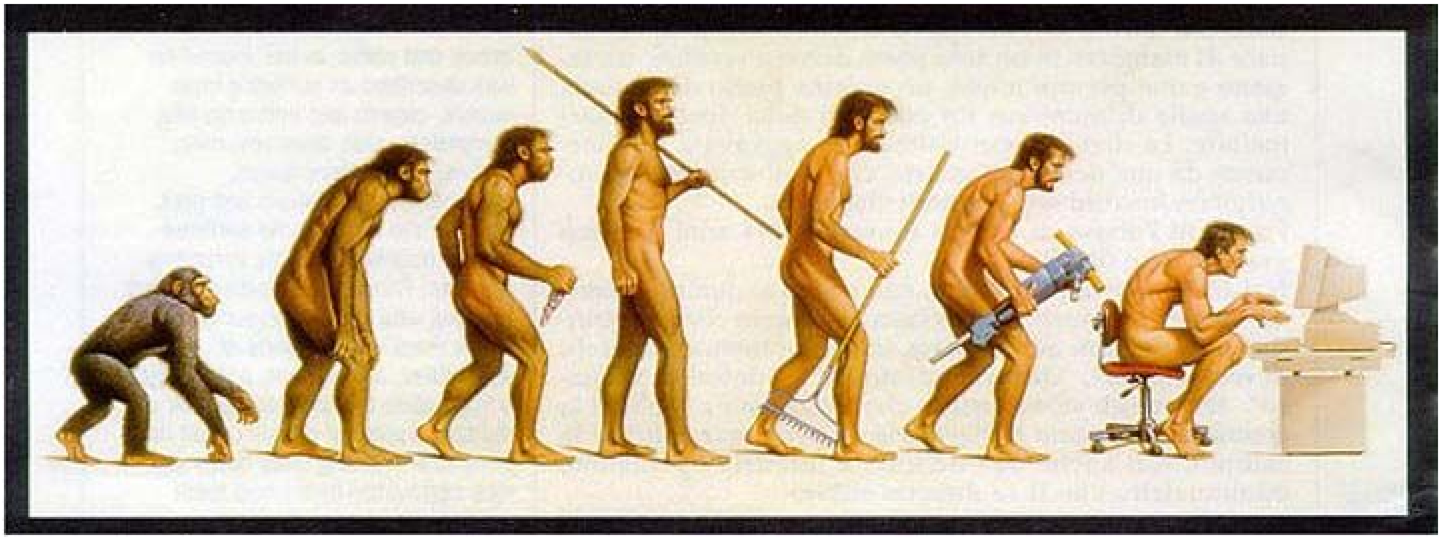
also behind the scenes: custom memory management, convolution library, efficient gpu array kernels, broadcasting, reduction, indexing, concatenation, gradients for all these...

Why Julia?

Why Julia?

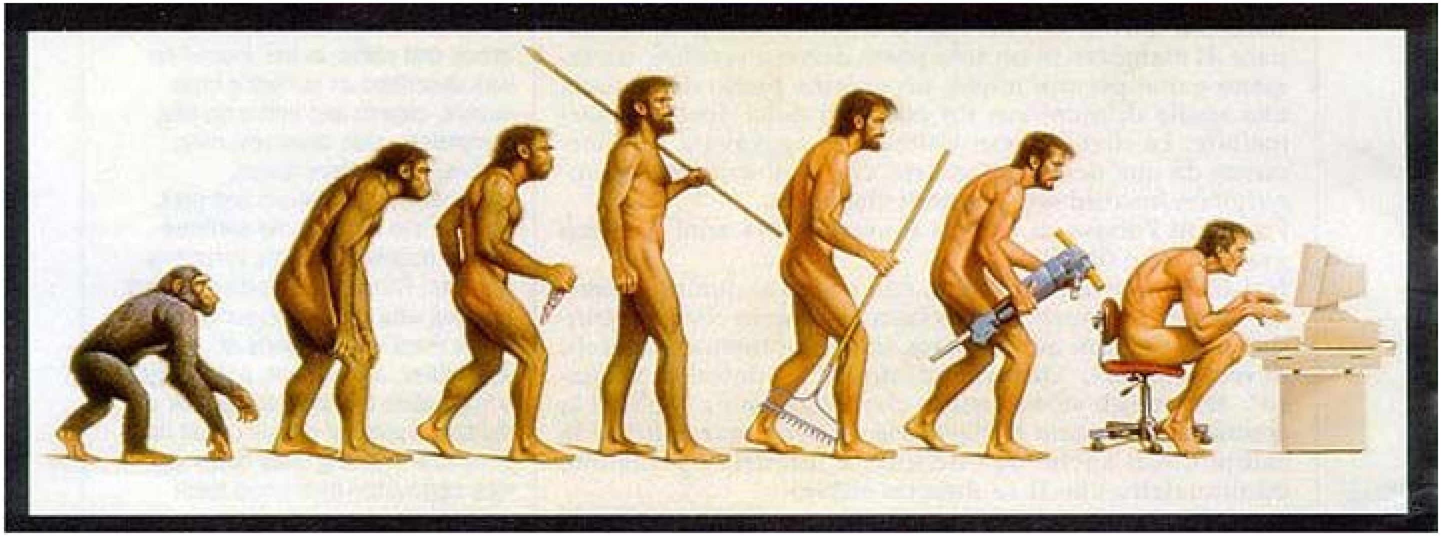
Efficiency   Expressivity

Evolution of computer languages



Machine Code | Assembler | Fortran BASIC | C → Julia deep learning

Evolution of ~~computer~~ languages



Machine Code

Assembler

Fortran BASIC

C

Julia

CUDA ConvNet

Caffe

Torch Theano TFlow

PyTorch DyNet TF Fold

Machine Learning in 5 Slides

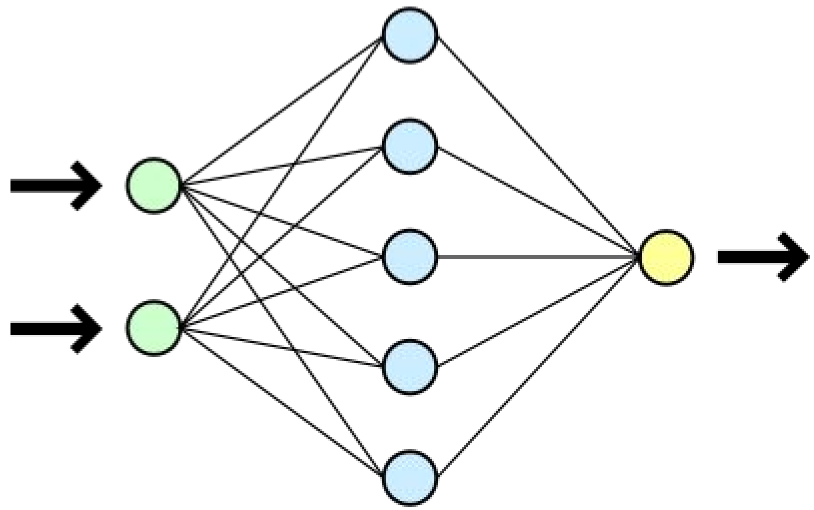
Machine learning: observations

Machine learning: modeling

Inputs

x_1
x_2
.
.
.
x_n

Differentiable Program



Pred.

$\hat{y}_1$
$\hat{y}_2$
.
.
.
$\hat{y}_n$

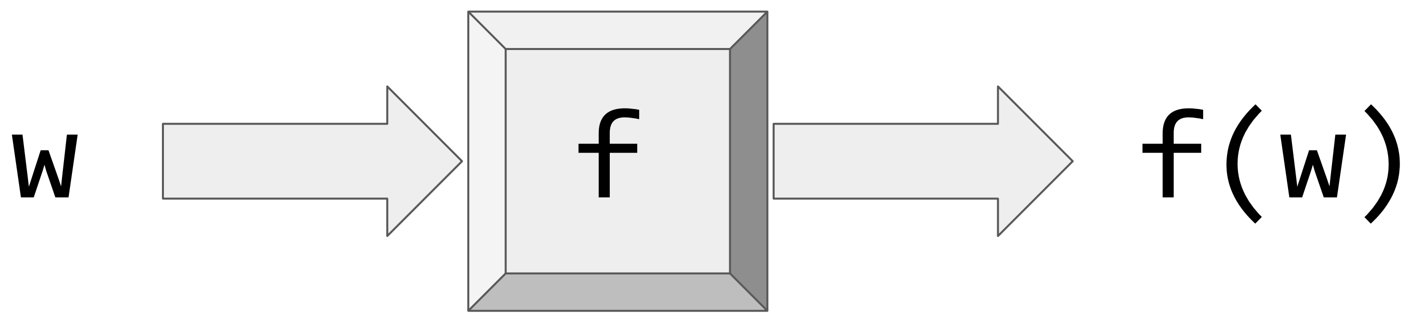
Machine learning: loss (error) function

```
function loss(w, x, ygold)
    ypred = predict(w, x)
    return sum(abs2, ypred - ygold)
end
```

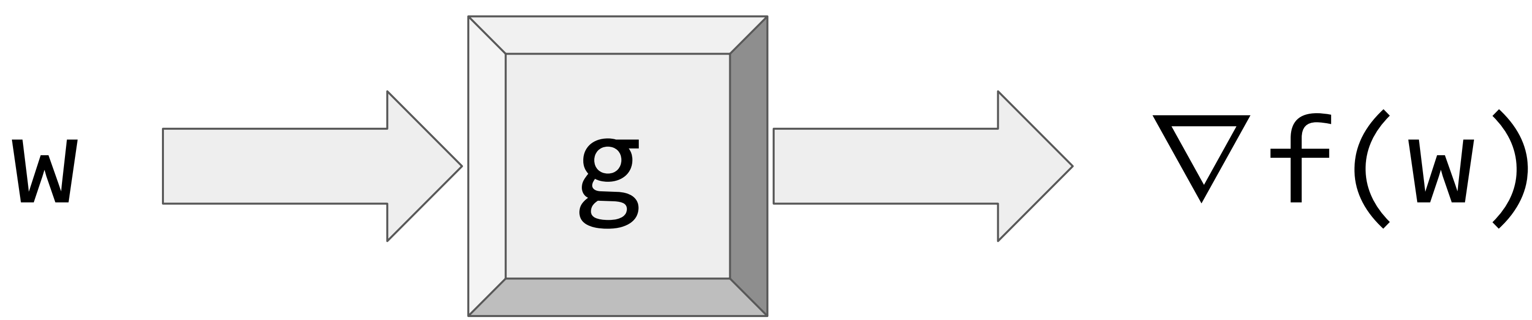
Machine learning: optimization loop

- Parameters w initialized randomly
- data = [(x1,y1),(x2,y2),...]: training examples
- loss(w,x,y): badness of predictions using w
- gfun(w,x,y): gradient of loss wrt w
- SGD(w,data,loss): find w that minimize loss

```
function SGD(w, data, loss)
    gfun = grad(loss)
    for (x,y) in data
        g = gfun(w, x, y)
        w = w - g * learningRate
    end
    return w
end
```




```
g = grad(f)
```



- Need functional code, no array overwriting.
- Most Julia functions supported.

Division of labor in machine learning

Programmer collects data, determines the model, its parameters, and its loss function.

Machine learning framework (Knet, TensorFlow etc.) optimizes parameters using the gradient of the loss function.

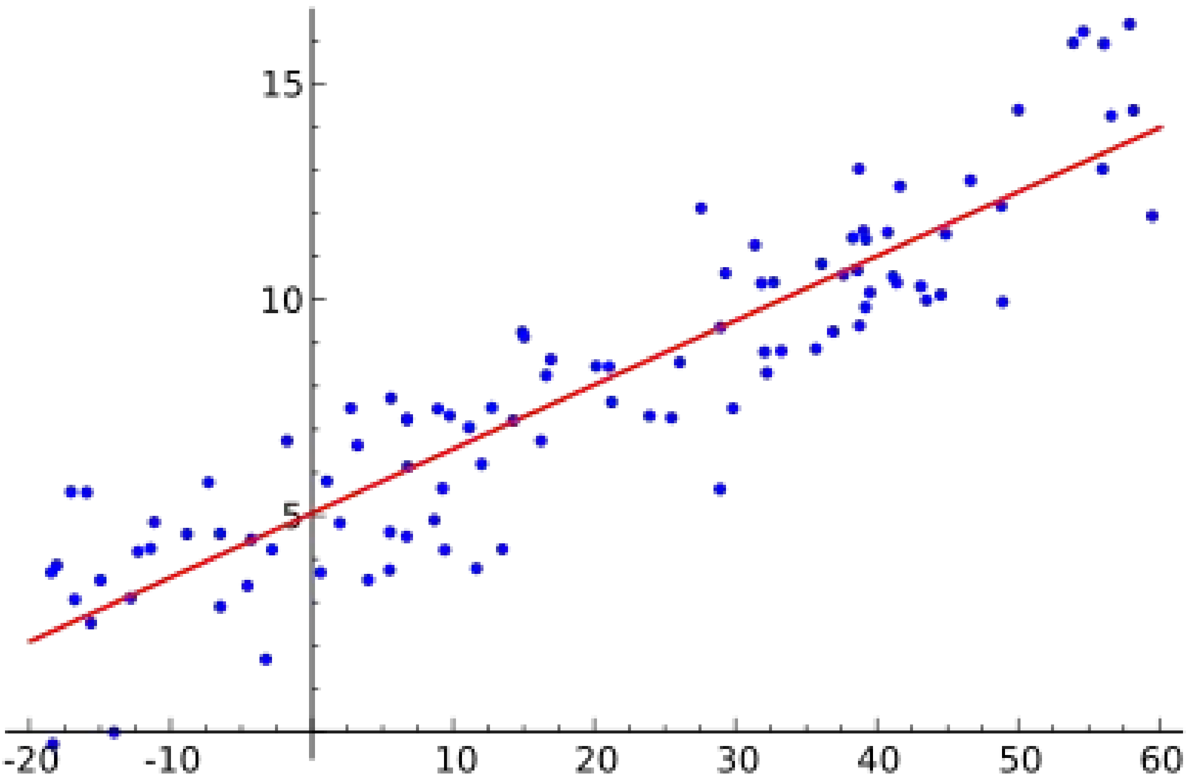
5 example models

Linear Regression

Linear regression: data

- "housing.data" from UCI ML repository.
- Real estate prices for 506 neighborhoods in Boston.
- Inputs: crime rate, average room number, distance to job centers etc. (13 features).
- Output: average house value.

Linear regression: model



$$y = wx + b$$

Linear regression: code

```
predict(w,x) = w[1]*x .+ w[2]

loss(w,x,y) = sum(abs2, y-predict(w,x))

lossgradient = grad(loss)
```

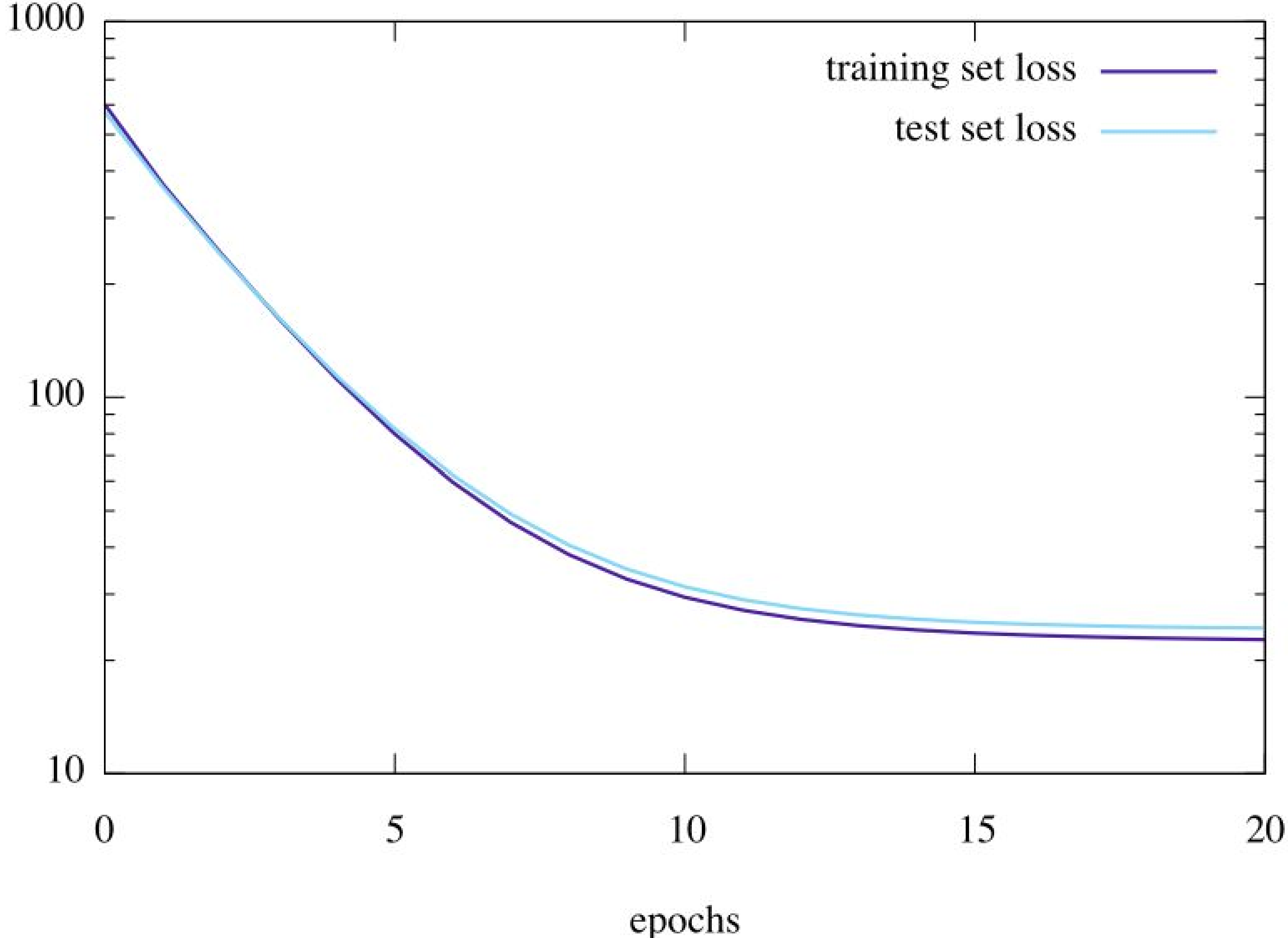
Linear regression: training

```
function train(w, data; lr=.1)
    for (x,y) in data
        dw = lossgradient(w, x, y)
        for i in 1:length(w)
            w[i] -= lr * dw[i]
        end
    end
    return w
end
```

Linear regression: sample run

```
[Knet/examples]$ julia housing.jl
# housing.jl (c) Deniz Yuret, 2016. Linear regression model for
# the Housing dataset from the UCI Machine Learning Repository.

opts=(:seed,-1)(:epochs,20)(:lr,0.1)(:atype,"Array")(:fast,false)

size(data) = (14,506)

(:epoch,0,:trn,601.6809326430163,:tst,574.1631980915728)
(:epoch,1,:trn,369.1969797775598,:tst,362.5340494097422)
(:epoch,2,:trn,241.53609243589833,:tst,239.26355906515374)
...
(:epoch,18,:trn,22.9400510062471,:tst,24.56666584413355)
(:epoch,19,:trn,22.815344401770275,:tst,24.457955035865993)
(:epoch,20,:trn,22.717508849652422,:tst,24.37935078764147)

  2.095076 seconds (686.43 k allocations: 30.067 MB, 0.70% gc time)
```

Linear regression: learning curve

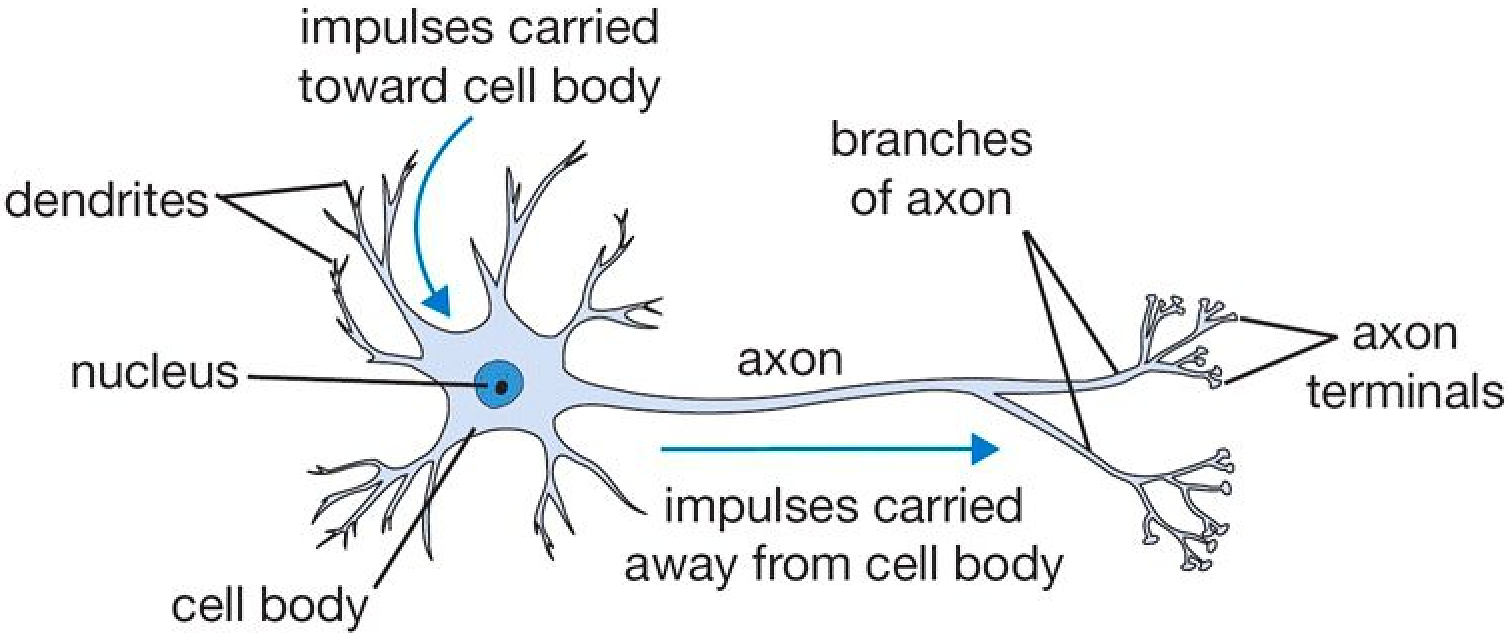


Softmax Classification

Softmax classification: data

- Dataset: MNIST
- Inputs: 28x28 handwritten digits, 60000 examples
- Output: Digit category: 0..9

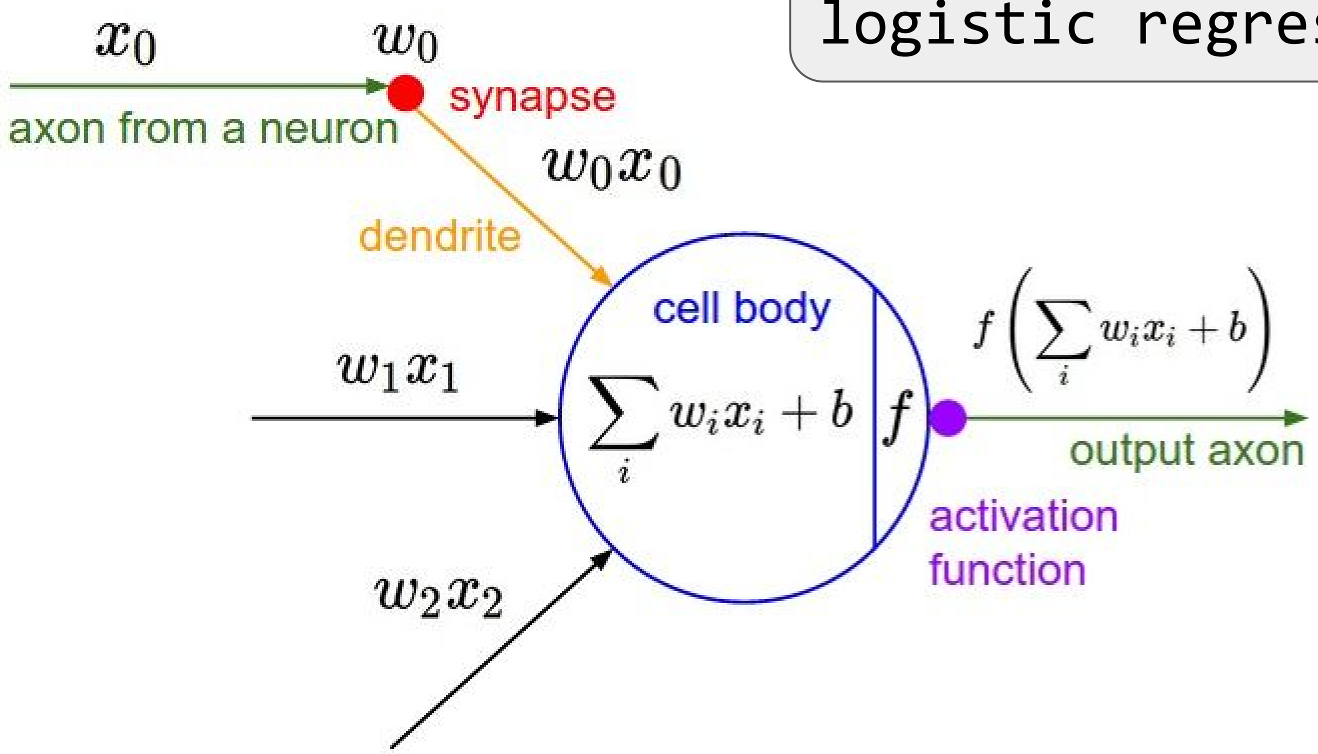
Softmax classification: model



http://cs231n.github.io/neural-networks-1/

Softmax classification: model

AKA perceptron, logistic regression



http://cs231n.github.io/neural-networks-1/

Softmax classification: code

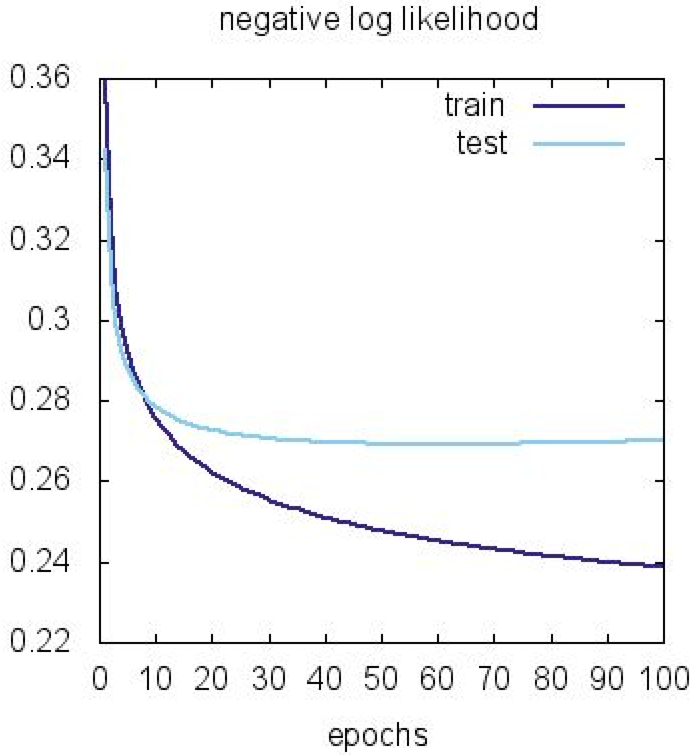
Cross entropy loss = $-\sum p_i \ln \hat{p}_i$

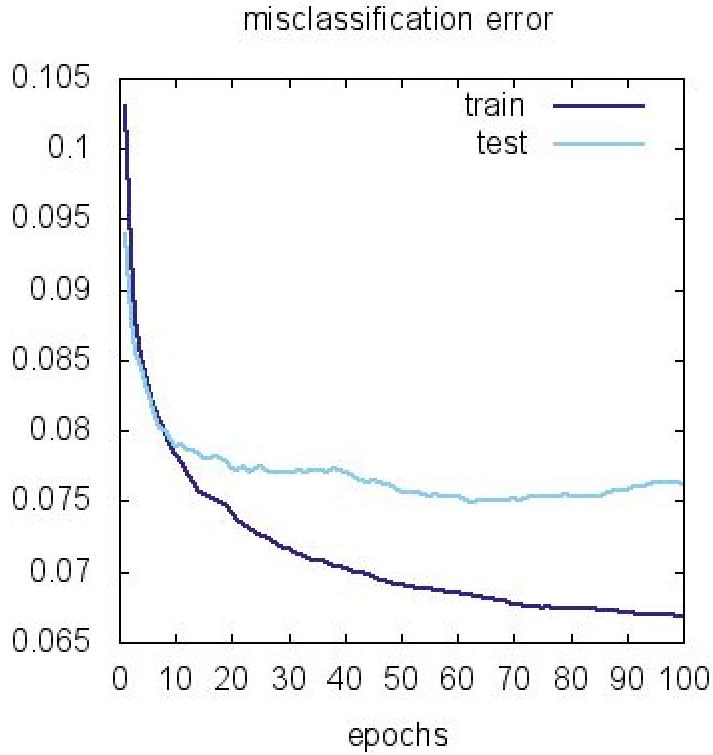
```
function loss(w,x,ygold)
    ypred = predict(w,x)
    ynorm = ypred .- log.(sum(exp.(ypred),1))
    -sum(ygold .* ynorm) / size(ygold,2)
end

predict(w,x) = w[1]*x .+ w[2]     # same as linear

lossgradient = grad(loss)         # same as linear

function train()...               # same as linear
```

Softmax classification: sample run

```
[Knet/examples]$ julia mnist.jl
# Handwritten digit recognition problem from
# http://yann.lecun.com/exdb/mnist.

INFO: Loading MNIST...
opts=(:seed,-1)(:batchsize,100)(:hidden,[])
(:epochs,10)(:lr,0.5)(:winit,0.1)(:fast,false)

(:epoch,0,:trn,0.08575,:tst,0.0807)
(:epoch,1,:trn,0.8991666666666667,:tst,0.9036)
 ...
(:epoch,9,:trn,0.9187666666666666,:tst,0.9154)
(:epoch,10,:trn,0.91945,:tst,0.9154)
```

Softmax classification: learning curve



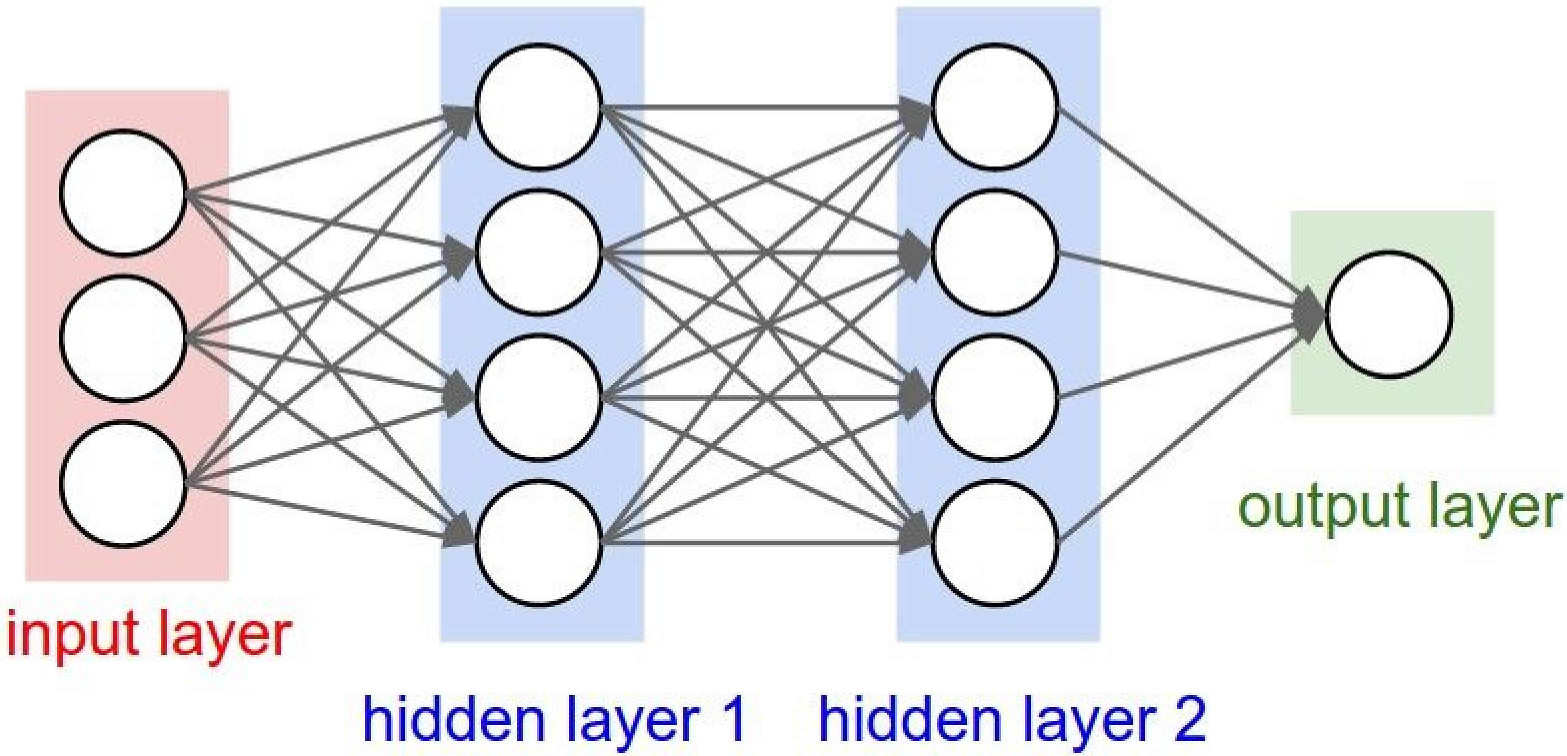


Multilayer Perceptrons

Multilayer perceptrons: data

- Dataset: MNIST
- Inputs: 28x28 handwritten digits, 60000 examples
- Output: Digit category: 0..9

Multilayer perceptrons: model



http://cs231n.github.io/neural-networks-1/

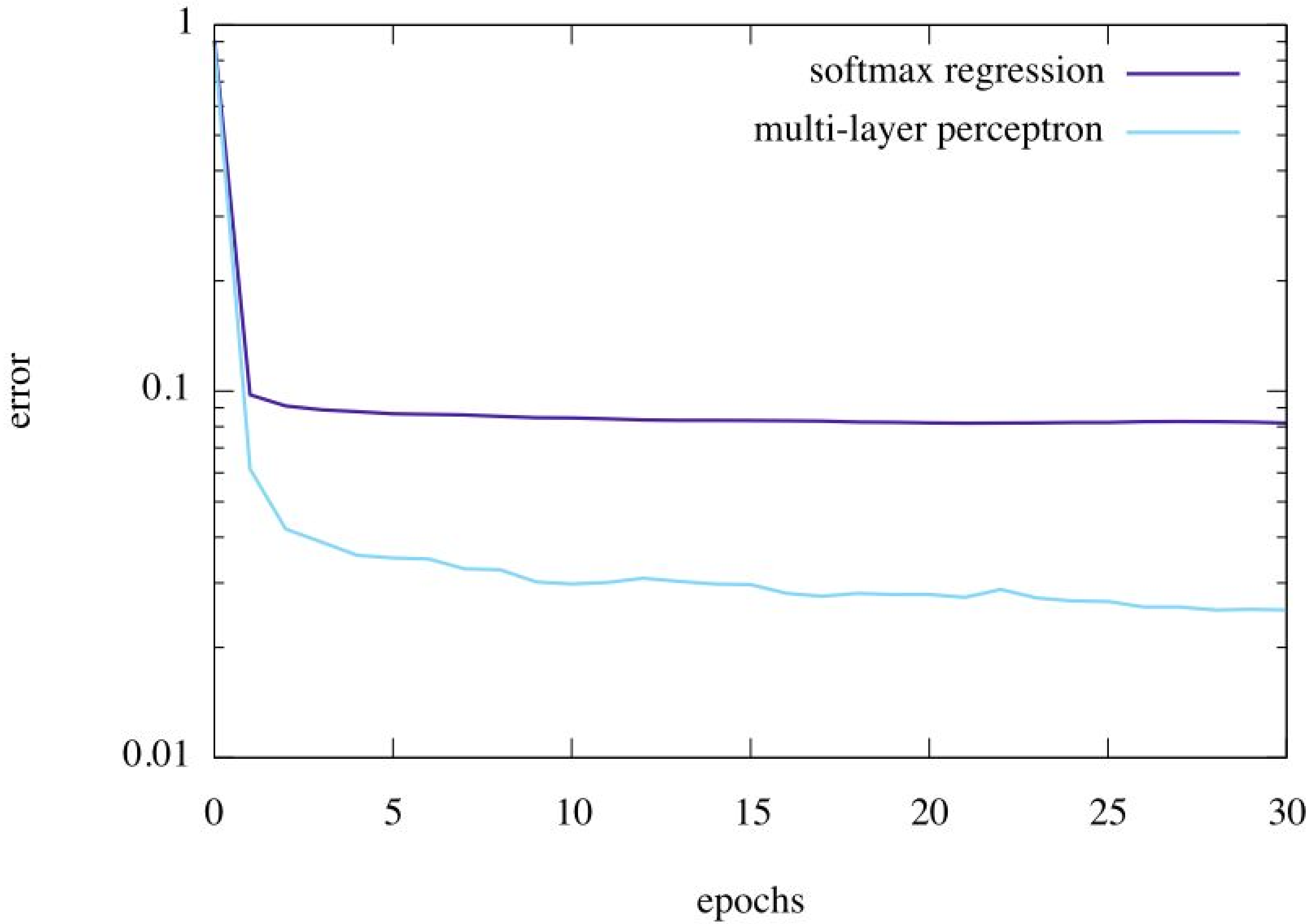
Multilayer perceptrons: code

```
function predict(w,x)
    for i=1:2:length(w)
        x = w[i]*x .+ w[i+1]
        if i<length(w)-1
            x = max.(0,x)
        end
    end
    return x
end

function loss(w,x,ygold)...     # same as softmax

lossgradient = grad(loss)       # same as softmax

function train()...             # same as softmax
```

Multilayer perceptrons: sample run

```
[Knet/examples]$ julia mnist.jl --hidden 64
# Handwritten digit recognition problem from
# http://yann.lecun.com/exdb/mnist.

INFO: Loading MNIST...
opts=(:seed,-1)(:batchsize,100)(:hidden,[])
(:epochs,10)(:lr,0.5)(:winit,0.1)(:fast,false)

(:epoch,0,:trn,0.1094,:tst,0.1071)
(:epoch,1,:trn,0.9435,:tst,0.9424)
 ...
(:epoch,9,:trn,0.9863,:tst,0.9728)
(:epoch,10,:trn,0.9875,:tst,0.9724)
```

Multilayer perceptrons: learning curve

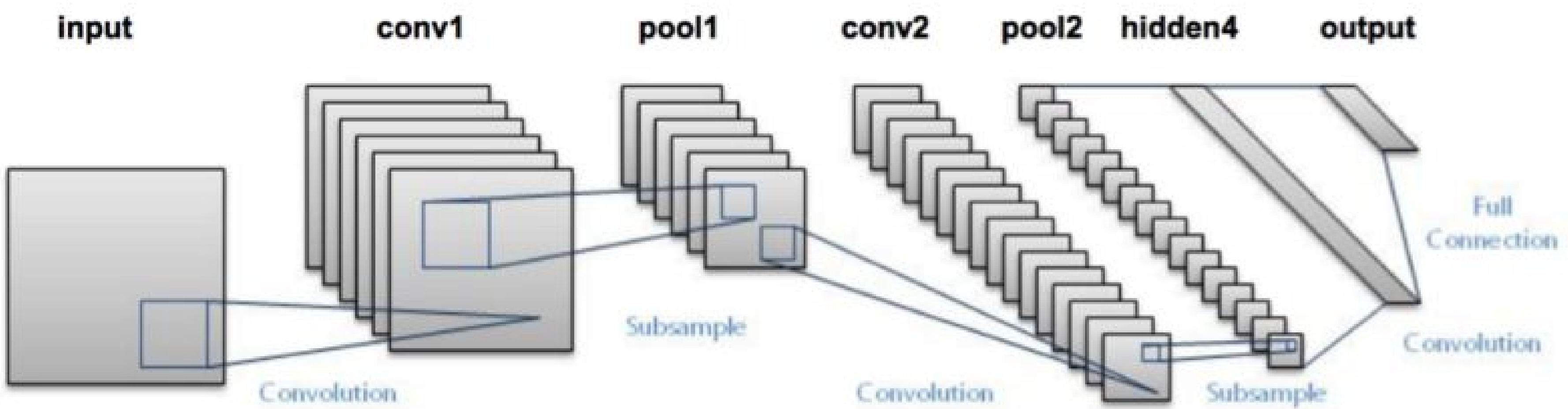


Convolutional Neural Networks

Convolutional neural networks: data

- Dataset: MNIST
- Inputs: 28x28 handwritten digits, 60000 examples
- Output: Digit category: 0..9

Convolutional neural networks: model



http://cs231n.github.io/convolutional-networks/

Convolutional neural networks: code

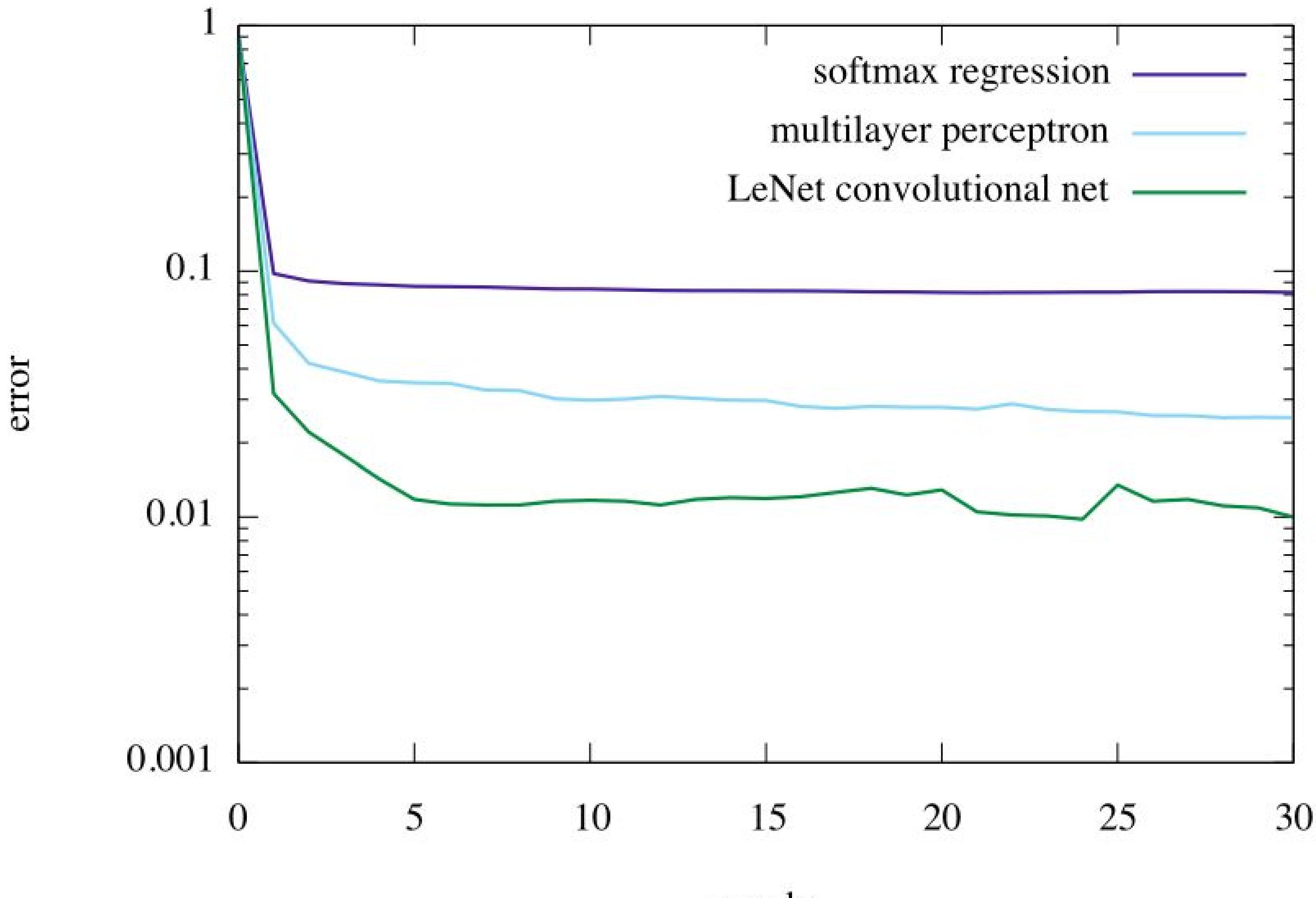
```
function predict(w,x,n=length(w)-4) # LeNet model
    for i=1:2:n
        x = pool(relu.(conv4(w[i],x) .+ w[i+1]))
    end
    for i=n+1:2:length(w)-2
        x = relu.(w[i]*mat(x) .+ w[i+1])
    end
    return w[end-1]*x .+ w[end]
end

function loss(w,x,ygold)...    # same as softmax

lossgradient = grad(loss)      # same as softmax

function train()...            # same as softmax
```

Compare with Caffe implementation:
https://github.com/BVLC/caffe/blob/master/examples/mnist/lenet.prototxt

Convolutional neural networks: sample run

```
[Knet/examples]$ julia mnist4d.jl
INFO: Loading MNIST
INFO: Testing lenet (convolutional net) on MNIST
("epochs"=>100,"lr"=>0.1,"seed"=>42,"gcheck"=>0,
"batchsize"=>100)

(1,0.9656,0.9683,...
(2,0.9774,0.9779,...
 ...
(9,0.9950,0.9884,...
(10,0.9957,0.9883,...
```

Convolutional neural networks: learning curve



Recurrent Neural Networks

Recurrent neural networks: data

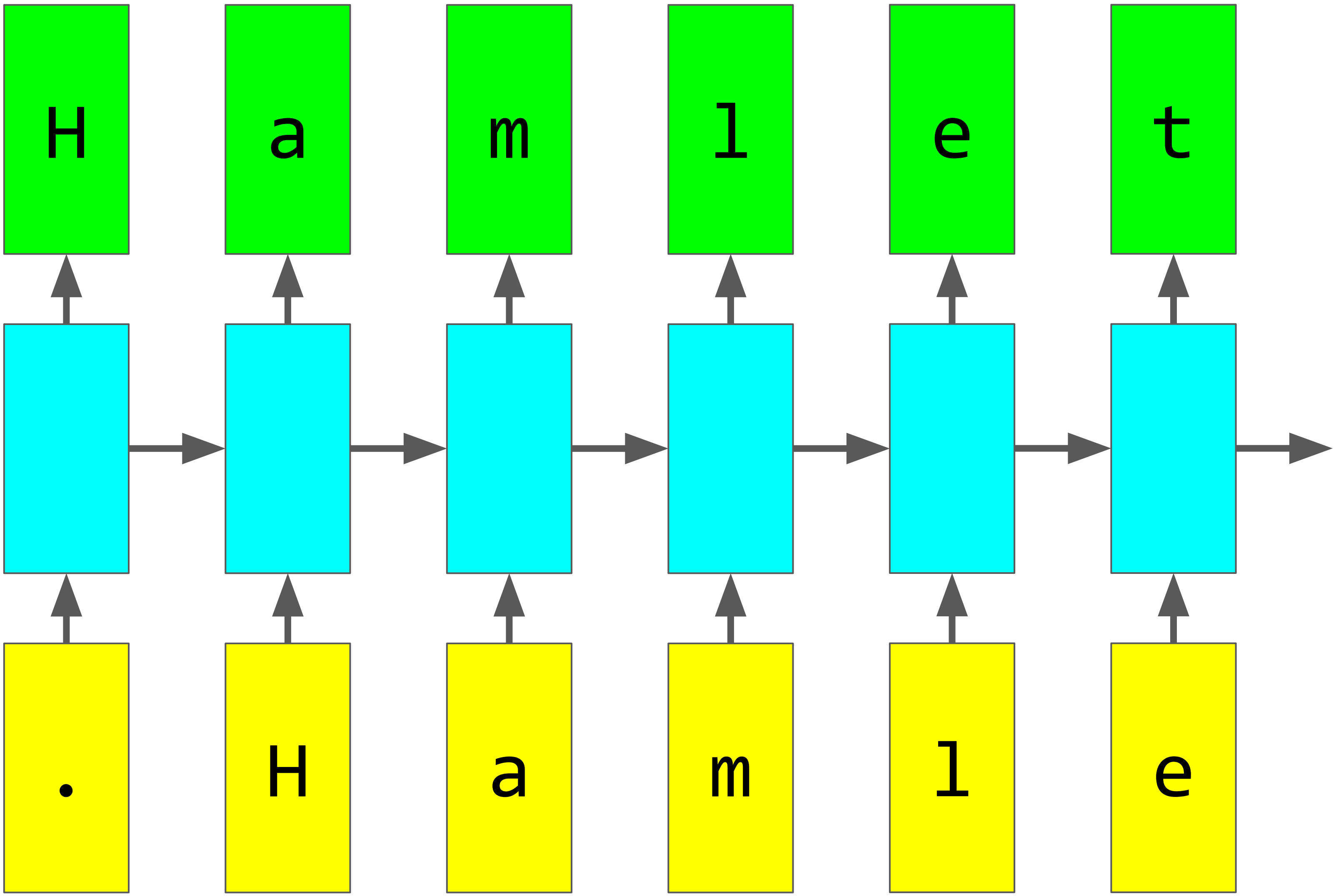
The Complete Works of William Shakespeare

```
  ...
  Edm. It is his hand, my lord; but I hope his heart is not in the
     contents.
  Glou. Hath he never before sounded you in this business?
  Edm. Never, my lord. But I have heard him oft maintain it to be fit
     that, sons at perfect age, and fathers declining, the father
     should be as ward to the son, and the son manage his revenue.
  Glou. O villain, villain! His very opinion in the letter! Abhorred
     villain! Unnatural, detested, brutish villain! worse than
     brutish! Go, sirrah, seek him. I'll apprehend him. Abominable
     villain! Where is he?
  ...
```

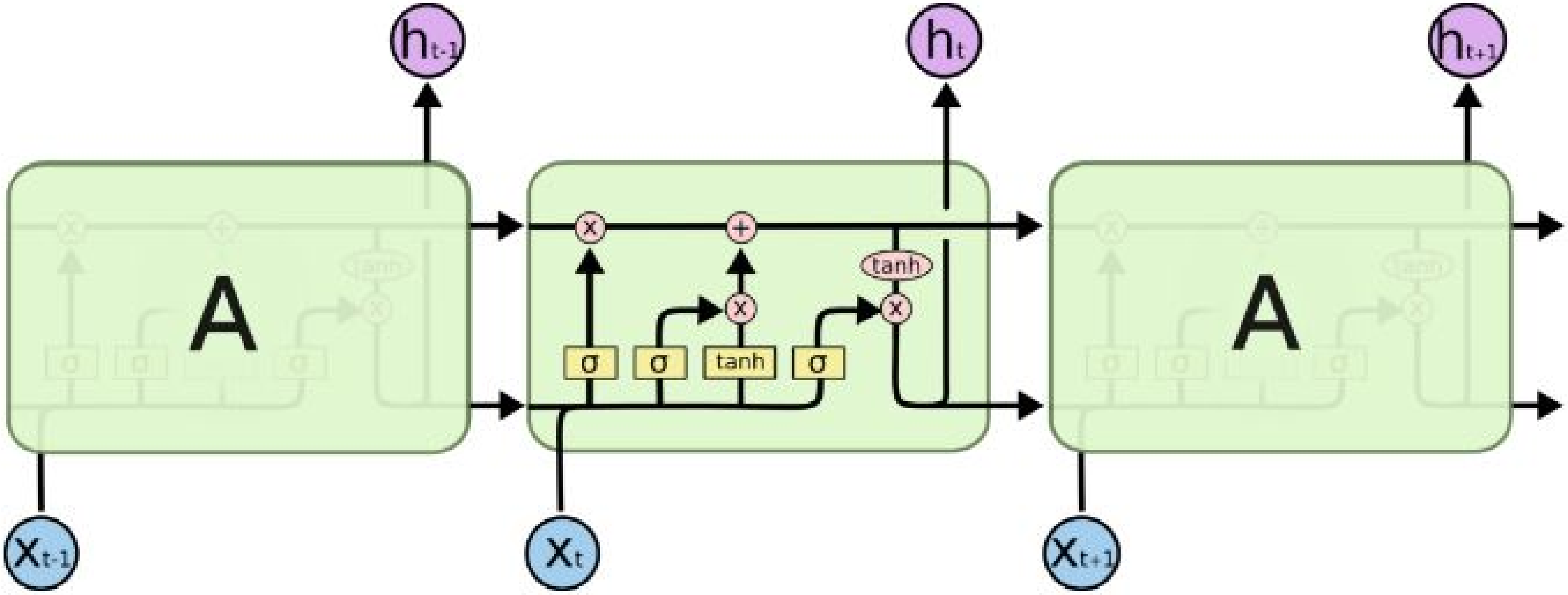
(5589887 characters)

http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Recurrent neural networks: model



Recurrent neural networks: model



Long short term memory (LSTM) modules

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Recurrent neural networks: model

$$
\begin{aligned}
f_t &= \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \\
i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \\
o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \\
c_t &= f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \\
h_t &= o_t \circ \sigma_h(c_t)
\end{aligned}
$$

Recurrent neural networks: code

```
function lstm(weight,bias,hidden,cell,input)
    gates   = hcat(input,hidden) * weight .+ bias
    h       = size(hidden,2)
    forget  = sigm(gates[:,1:h])
    ingate  = sigm(gates[:,1+h:2h])
    outgate = sigm(gates[:,1+2h:3h])
    change  = tanh(gates[:,1+3h:4h])
    cell    = cell .* forget + ingate .* change
    hidden  = outgate .* tanh(cell)
    return (hidden,cell)
end
```

Recurrent neural networks: sample run

```
[Knet7/examples]$ julia charlm.jl --data 100.txt
--epochs 100
(:lr=>1.0,:dropout=>0.0,:embedding=>256,:gclip=>5
.0,:hidden=>256,:epochs=>100,:nlayer=>1,:decay=>0
.9,:seqlength=>100,:seed=>42,:batchsize=>128)
INFO: Chars read: (5589917,)
INFO: 92 unique chars
(epoch,lr,loss)
(1,1.0,2.2447511306910757)
(2,1.0,1.5556333172749894)
(3,1.0,1.3716149988793005)
(4,1.0,1.288365624960702)
(5,1.0,1.2409912395974114)
```

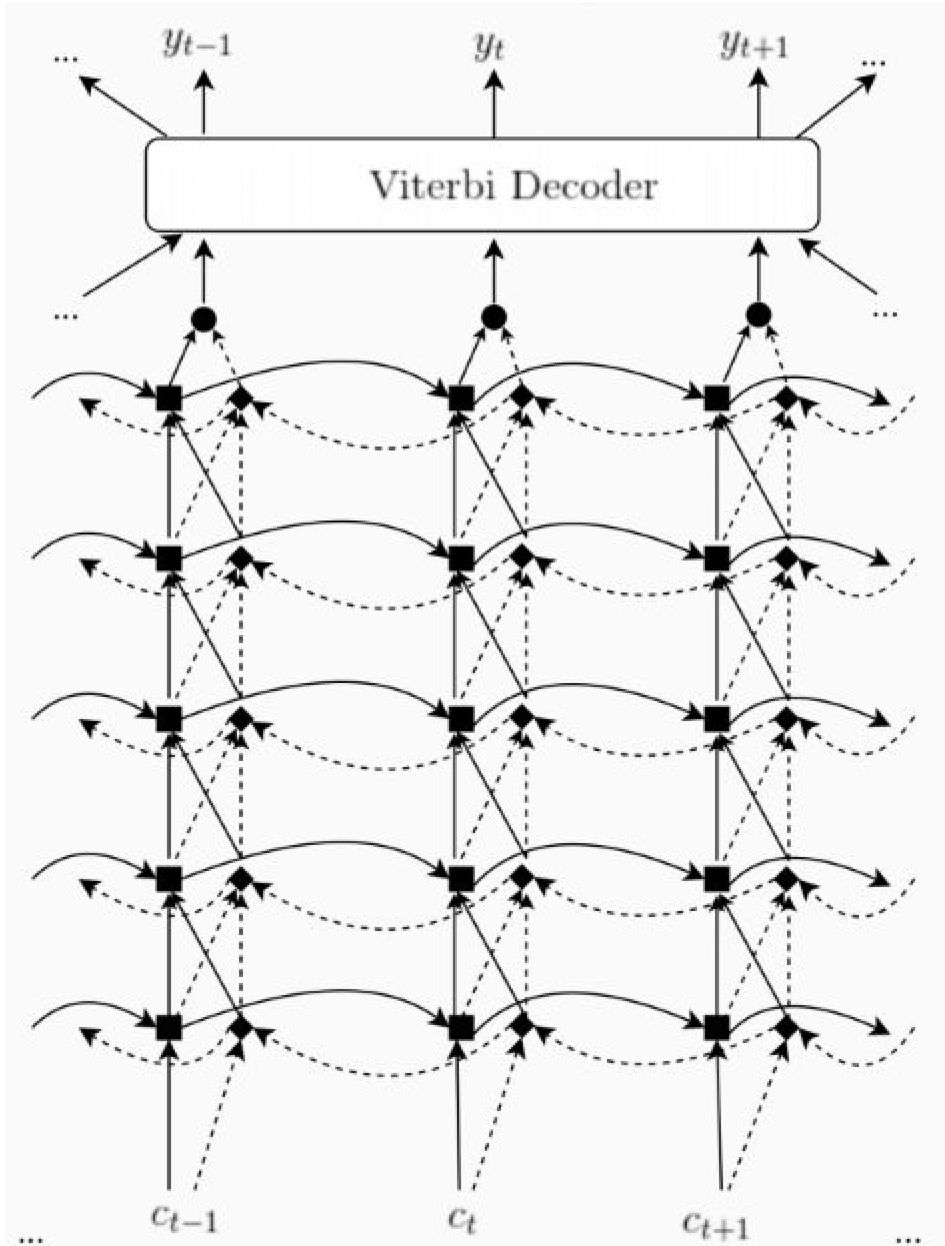
Recurrent neural networks: output

```
LUCETTA. Welcome, getzing a knot. There is as I thought you aim
  Cack to Corioli.
MACBETH. So it were timen'd nobility and prayers after God'.
FIRST SOLDIER. O, that, a tailor, cold.
DIANA. Good Master Anne Warwick!
SECOND WARD. Hold, almost proverb as one worth ne'er;
  And do I above thee confer to look his dead;
  I'll know that you are ood'd with memines;
  The name of Cupid wiltwite tears will hold
  As so I fled; and purgut not brightens,
  Their forves and speed as with these terms of Ely
  Whose picture is not dignitories of which,
  Their than disgrace to him she is.
...
```

Composing jazz with the same model...

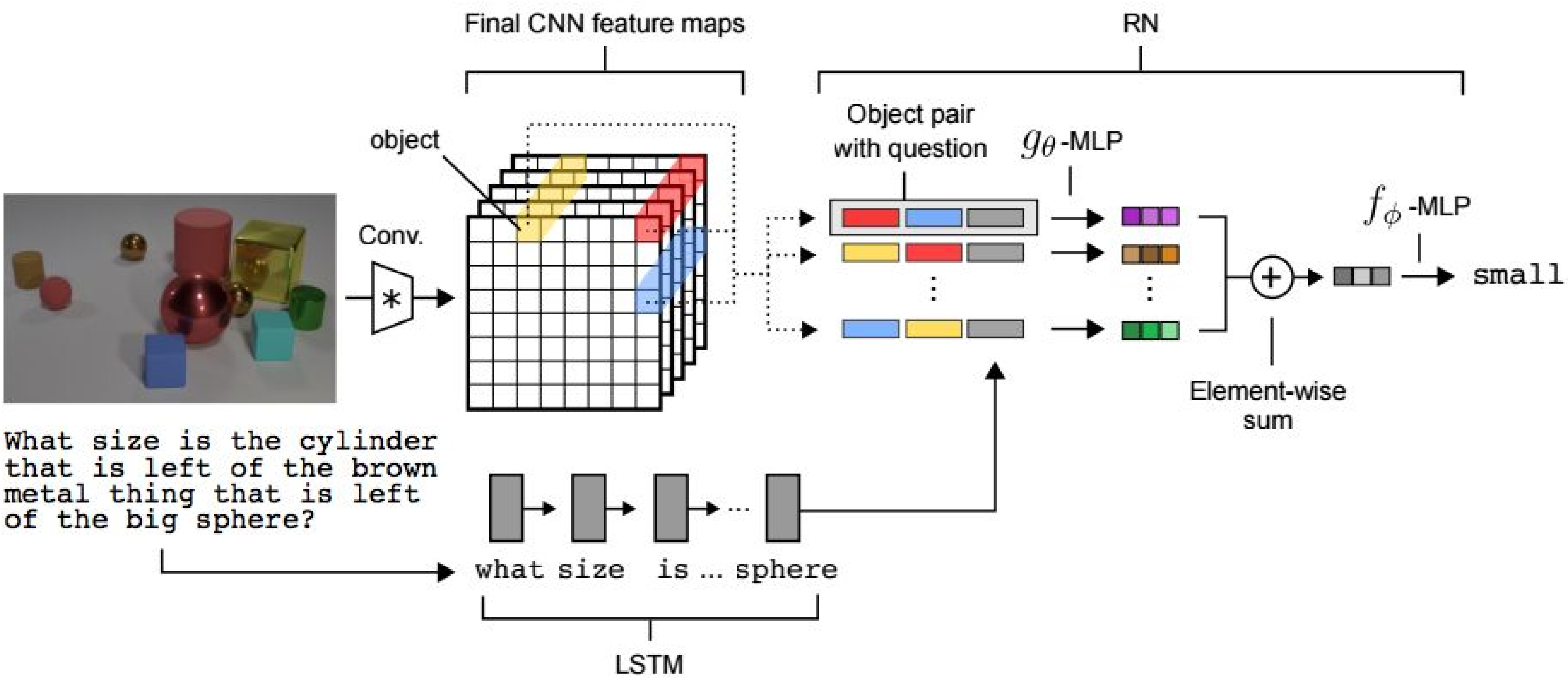
[LSTM Music]
by
Mustafa Ömer Gül

Onto more complex models...



Onur Kuru (2016). Character-level tagging. MS Thesis. Koç University.

Onto more complex models...



https://arxiv.org/abs/1706.01427

Under the hood

KnetArray
(yet another GPU array type)

KnetArray: usage

```
# Construction and conversion
KnetArray{T}(dims)
KnetArray(a::AbstractArray)
Array(k::KnetArray)

# Machine learning
w = initmodel()
x = initdata()
train(w, x)   # runs on cpu
kw = map(KnetArray, w)
kx = map(KnetArray, x)
train(kw, kx) # runs on gpu
```

KnetArray features

- **custom memory manager**
  minimizes the number of calls to the slow cudaMalloc by reusing already allocated but garbage collected GPU pointers.
- **custom kernels**
  implement elementwise, broadcasting, reduction, indexing, concatenation etc. operations.
- **custom getindex**
  handles ranges such as a[5:10] as views with shared memory instead of copies.

AutoGrad.jl

dynamic computational graph based
automatic differentiation
of (almost all of) Julia
(including getindex and cat)

Differentiation techniques

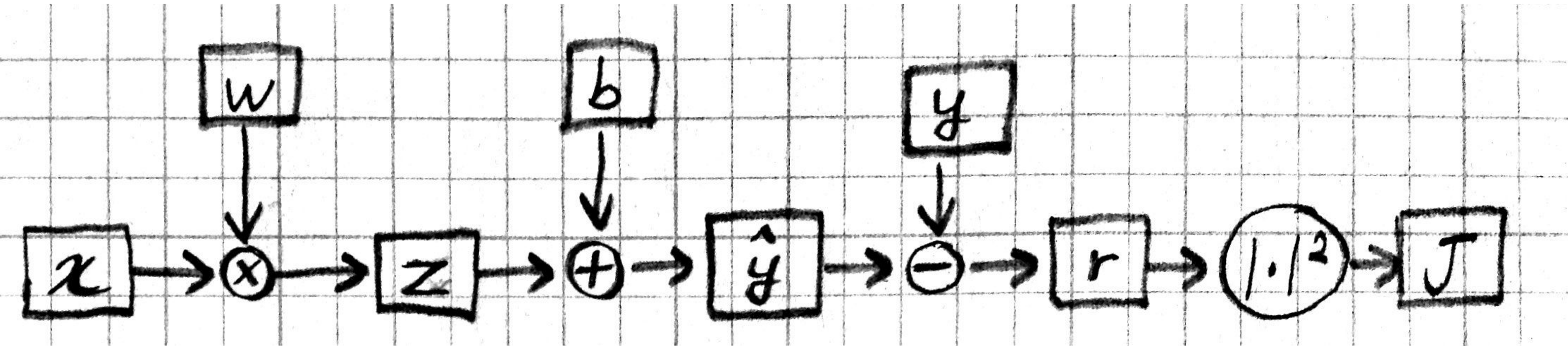
- Manual programming
- Numerical approx: $f'(x) \approx \frac{f(x+\varepsilon)-f(x-\varepsilon)}{2\varepsilon}$
- Symbolic derivative: Maxima, Mathematica, Maple
- Static graph methods: **Knet7**, Caffe, MXnet, Theano, Torch, TensorFlow, ...
- Dynamic graph methods: **Knet8**, autograd, Chainer, DyNet, PyTorch, TF-Fold

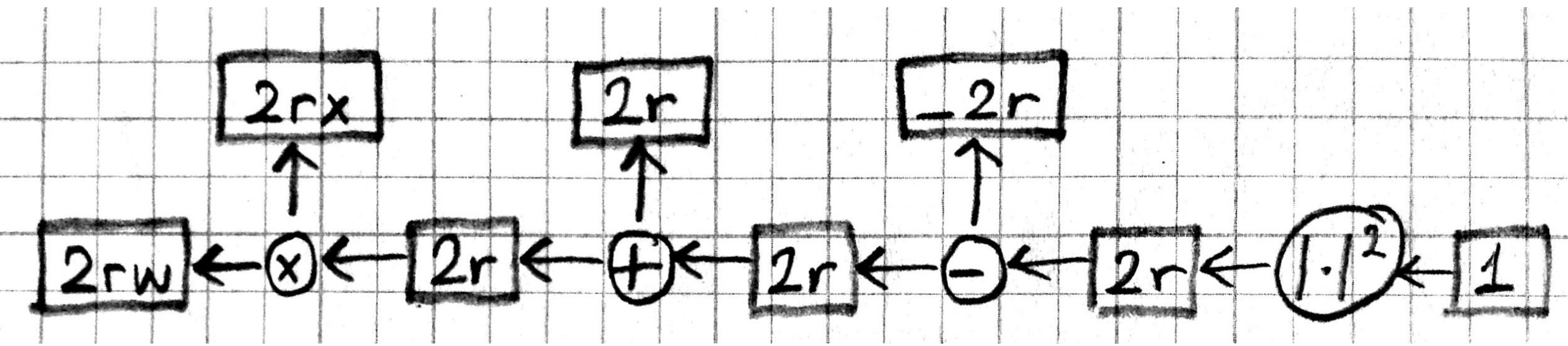
(*) [Atılım Güneş Baydin et al. (2015) Automatic differentiation in machine learning: a survey]

Manual programming

Linear regression example

$$\text{loss}(w,b,x,y) = |(wx+b) - y|^2$$





Model and loss function

```
function loss(w, x, ygold)
    ypred = w * x
    ydiff = ypred - ygold
    sqerr = ydiff .^ 2
    qloss = sum(sqerr)
end
```

Gradient of the loss function

```
function grad(w, x, ygold)
    ypred = w * x
    ydiff = ypred - ygold
    sqerr = ydiff .^ 2
    qloss = sum(sqerr)

    d_qloss = 1.0
    d_sqerr = d_qloss .* ones(sqerr)
    d_ydiff = d_sqerr .* (2 * ydiff)
    d_ypred = d_ydiff
    d_w     = d_ypred *  x'
end
```

Disadvantages of manual programming

- Requires too much effort and is error-prone.

Numerical approximation

Finite difference approximation

$$f'(x) \approx \frac{f(x+\varepsilon)-f(x-\varepsilon)}{2\varepsilon}$$

Disadvantages:

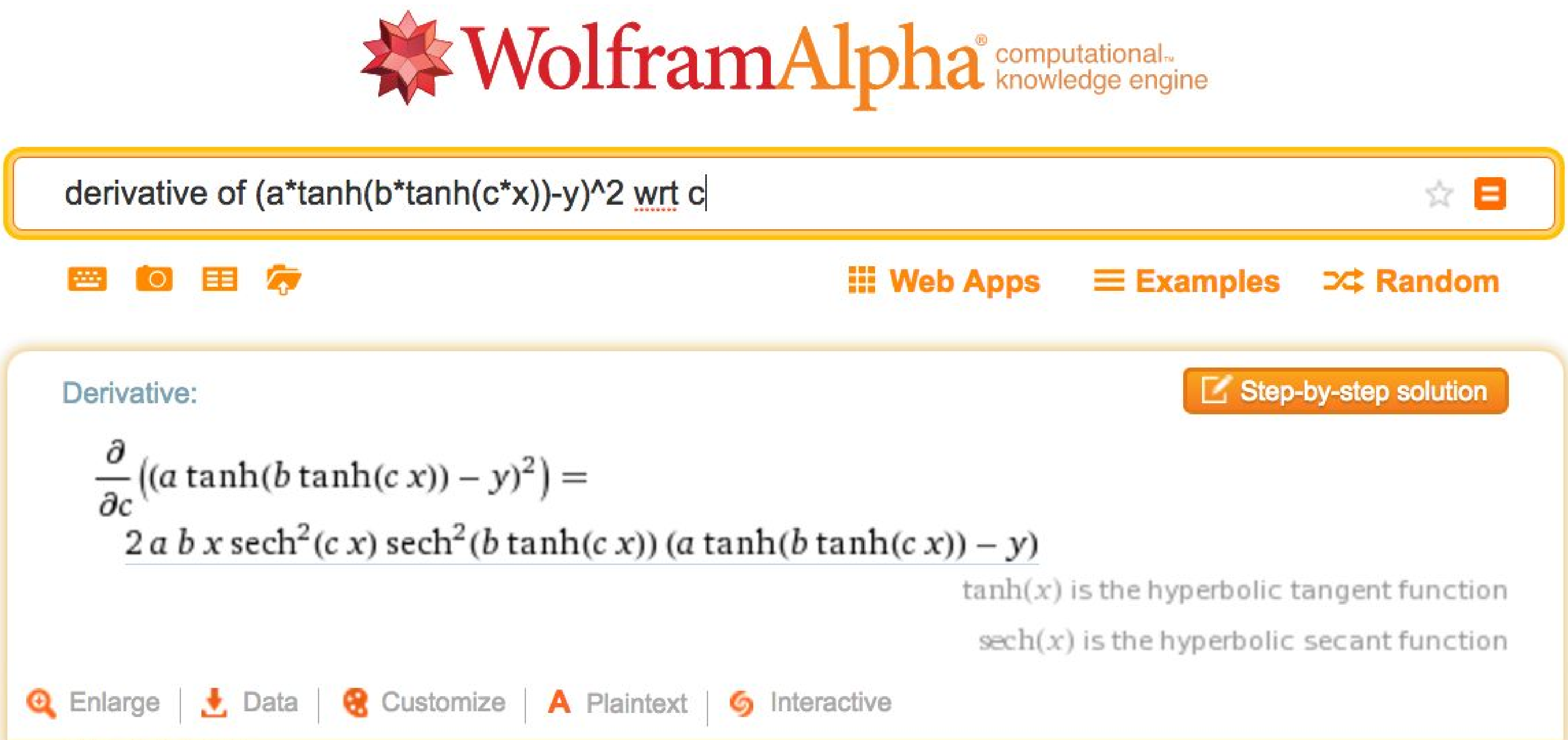
- If x has N elements, we need to compute this equation N times.

- It is difficult to know the right step size and numerical error can be large.

Symbolic derivatives
(Mathematica, Maple)

Working with algebraic expressions

A simple two layer network



Disadvantages of symbolic derivatives

- Difficult to handle high level programming constructs (loops, conditionals...).

- The resulting derivative expression can grow exponentially.

$$l_{n+1} = 4l_n(1 - l_n),\ l_1 = x$$

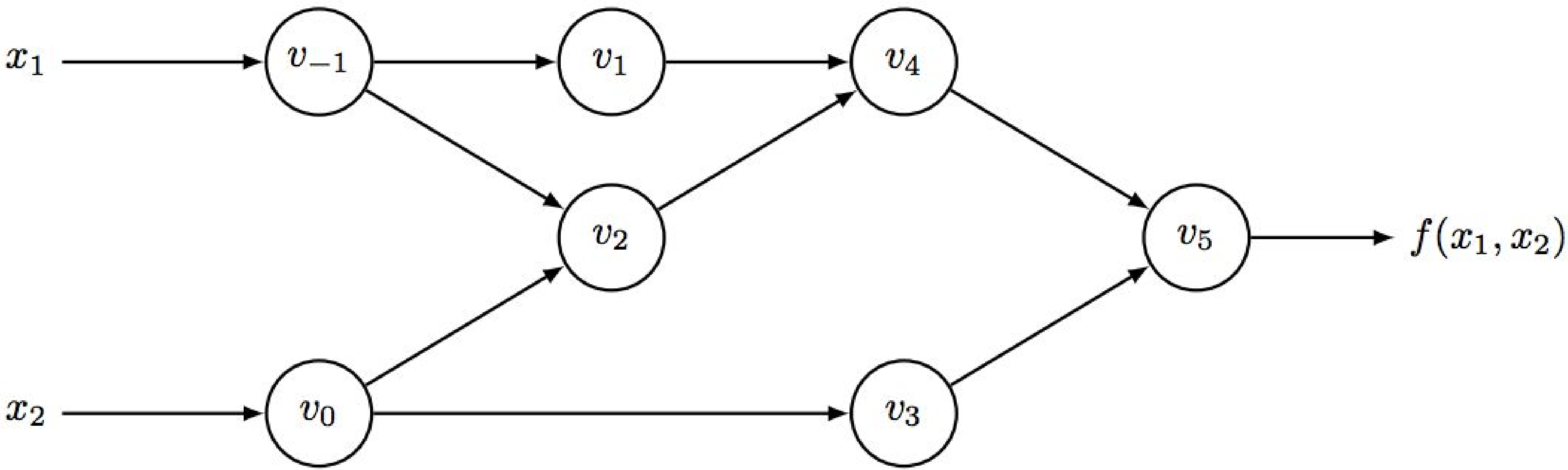
$$\frac{d}{dx} l_4 = 128x(1-x)(-8+16x)(1-2x)^2(1-8x+8x^2) + 64(1-x)(1-2x)^2(1-8x+8x^2)^2 - 64x(1-2x)^2(1-8x+8x^2)^2 - 256x(1-x)(1-2x)(1-8x+8x^2)^2$$

Static graph automatic diff.
(Knet1-7,Theano,Torch,Tflow…)

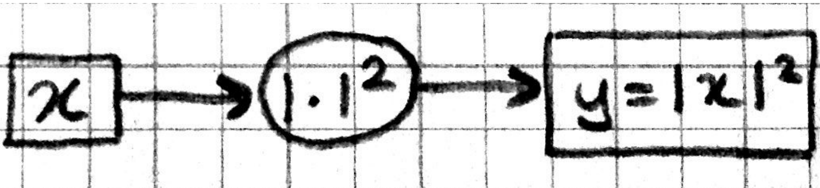
Construct the computational graph of fn

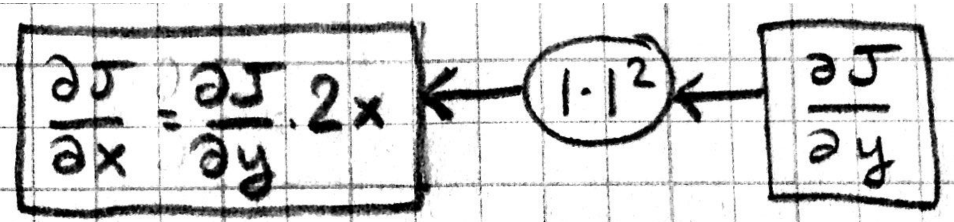
$$f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2)$$

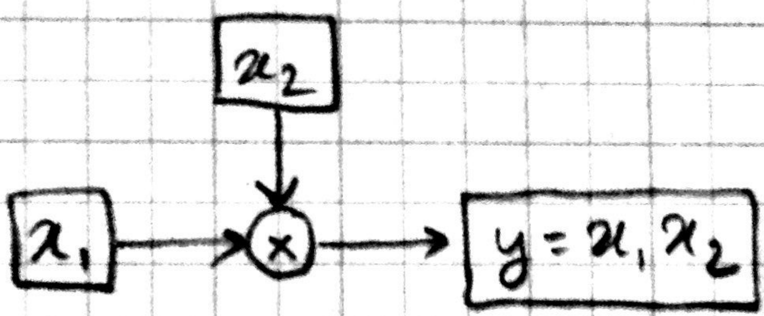
leads to the following graph:

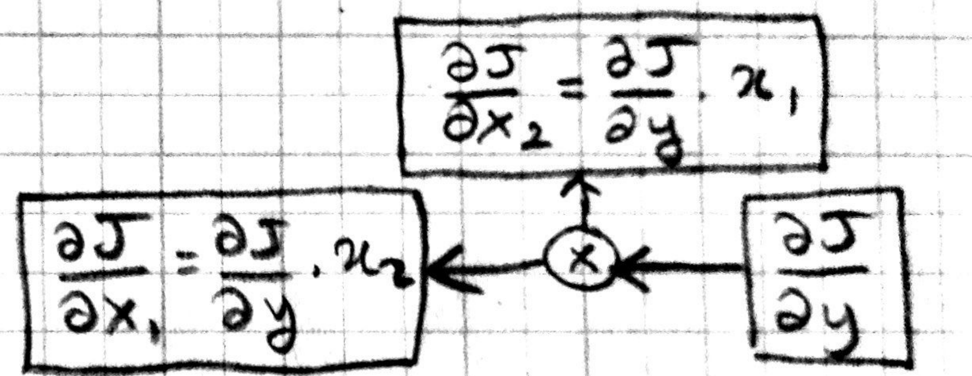


Define derivatives of primitive operations









Compute gradient of function wrt each variable going backward (reverse mode AD)

Forward Evaluation Trace		
$v_{-1} = x_1$		$= 2$
$v_0 = x_2$		$= 5$
$v_1 = \ln v_{-1}$		$= \ln 2$
$v_2 = v_{-1} \times v_0$		$= 2 \times 5$
$v_3 = \sin v_0$		$= \sin 5$
$v_4 = v_1 + v_2$		$= 0.693 + 10$
$v_5 = v_4 - v_3$		$= 10.693 + 0.959$
$y = v_5$		$= 11.652$

Reverse Adjoint Trace		
$\bar{\boldsymbol{x}}_1 = \bar{\boldsymbol{v}}_{-1}$		$\mathbf{= 5.5}$
$\bar{\boldsymbol{x}}_2 = \bar{\boldsymbol{v}}_0$		$\mathbf{= 1.716}$
$\bar{v}_{-1} = \bar{v}_{-1} + \bar{v}_1 \frac{\partial v_1}{\partial v_{-1}}$	$= \bar{v}_{-1} + \bar{v}_1 / v_{-1}$	$= 5.5$
$\bar{v}_0 = \bar{v}_0 + \bar{v}_2 \frac{\partial v_2}{\partial v_0}$	$= \bar{v}_0 + \bar{v}_2 \times v_{-1}$	$= 1.716$
$\bar{v}_{-1} = \bar{v}_2 \frac{\partial v_2}{\partial v_{-1}}$	$= \bar{v}_2 \times v_0$	$= 5$
$\bar{v}_0 = \bar{v}_3 \frac{\partial v_3}{\partial v_0}$	$= \bar{v}_3 \times \cos v_0$	$= -0.284$
$\bar{v}_2 = \bar{v}_4 \frac{\partial v_4}{\partial v_2}$	$= \bar{v}_4 \times 1$	$= 1$
$\bar{v}_1 = \bar{v}_4 \frac{\partial v_4}{\partial v_1}$	$= \bar{v}_4 \times 1$	$= 1$
$\bar{v}_3 = \bar{v}_5 \frac{\partial v_5}{\partial v_3}$	$= \bar{v}_5 \times (-1)$	$= -1$
$\bar{v}_4 = \bar{v}_5 \frac{\partial v_5}{\partial v_4}$	$= \bar{v}_5 \times 1$	$= 1$
$\bar{v}_5 = \bar{y}$	$= 1$	

Knet7 example computational graph

```
@knet function f(x1,x2)
    return relu(x1) + x1.*x2 - soft(x2)
end
```

```
julia> compile(:f)
1 Knet.Input()
2 Knet.Input()
3 Knet.Relu(1,)
4 Knet.Mul(1,2)
5 Knet.Add(3,4)
6 Knet.Soft(2,)
7 Knet.Axpb(6,)
8 Knet.Add(5,7)
```

Knet7 example primitive gradients

```
function back(::Sigm,y,dy,dx)
    for i = 1:length(dx)
        dx[i] = dy[i]*y[i]*(1-y[i])
    end
end

function back(::Relu,y,dy,dx)
    for i = 1:length(dx)
        dx[i] = dy[i] * (y[i] > 0)
    end
end
```

Disadvantages of static graph AD

- In order to statically compile models they are typically expressed in a restricted mini-language.

- These mini-languages typically have important features missing (e.g. array/dict indexing, concat, helper funcs, loops, conditionals...)

- The model is compiled once, therefore number/type/order of operations cannot change at run time.

[Maclaurin, D. et al. (2015). "Autograd: Effortless Gradients in Numpy."]

Disadvantages of static graph AD

- Your user manual has the phrase “computational graph”

  (a concept that should be relevant to compiler designers, not programmers)

Dynamic graph automatic diff.
(Knet8, Chainer, PyTorch…)

Differentiate all of Julia (or Python etc)

```julia
function f(x)
    s = 0
    for i = 1:length(x)
        if x[i] <= 0
            s += exp(x[i])
        else
            s += log(x[i])
        end
    end
    return s
end
```

```
julia> x = randn(3)'
 0.88  -2.52  1.55

julia> f(x)
0.3949414650701943

julia> 1./x
 1.13  -0.39  0.64

julia> exp(x)
 2.41  0.08  4.71

julia> g=grad(f); g(x)
 1.13  0.08  0.64
```

How does dynamic graph AD grad(f) work?

- While the user function f is running, primitive operations are recorded with inputs/outputs.
- When the program ends, the recorded operations constitute a (dynamically constructed) computational graph which is used for automatic differentiation.
- Since there is no pre-compilation (i) f can be written in a high-level language, (ii) f can change its graph based on its inputs.

Does recording slow things down?

```
operation            time
----------           ----
a1 = w1 * x          0.56
a2 = a1 .+ b1        0.59
a3 = max(0,a2)       0.62
a4 = w2 * a3         0.75
a5 = a4 .+ b2        0.78
a6 = a5 - y          0.82
a7 = a6 .^ 2         0.85
a8 = sum(a7)         1.06
recording            1.18
backprop             2.10
```

11%

Advantages of dynamic graph AD

All high level language features can be used when defining and training a model:

- loops
- conditional expressions
- helper functions
- recursive functions
- high-level functions and closures
- . . .

Knet repo and docs

[github.com/denizyuret/Knet.jl]()

[denizyuret.github.io/Knet.jl]()

[knet-users mailing list]()