# Deep neural networks II

May 30th, 2019

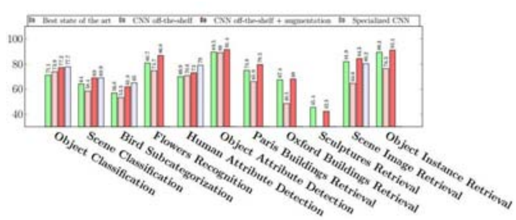
Yong Jae Lee
UC Davis

Many slides from Rob Fergus, Svetlana Lazebnik, Jia-Bin Huang, Derek Hoiem, Adriana Kovashka, Andrej Karpathy

## Why (convolutional) neural networks?

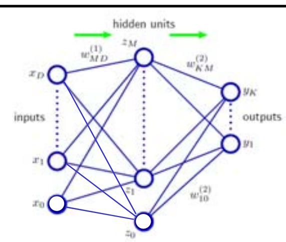
State of the art performance on many problems

Most (all?) papers in recent vision conferences use deep neural networks




Razavian et al., CVPR 2014 Workshops

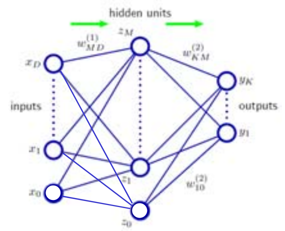
## Neural network definition




- *Nonlinear* classifier
- Can approximate any continuous function to arbitrary accuracy given sufficiently many hidden units

Figure from Christopher Bishop

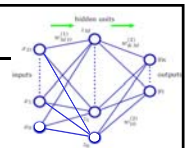
## Neural network definition




- Activations: $a_j = \sum_{i=0}^{D} w_{ji}^{(1)} x_i$
- Nonlinear activation function *h* (e.g. sigmoid, ReLU): $z_j = h(a_j)$

Figure from Christopher Bishop

## Neural network definition



- Layer 2

$$a_j = \sum_{i=0}^{D} w_{ji}^{(1)} x_i$$

$$z_j = h(a_j)$$

- Layer 3 (final)

$$a_k = \sum_{j=0}^{M} w_{kj}^{(2)} z_j$$

- Outputs (e.g. sigmoid/softmax)

(binary) $y_k = \sigma(a_k) = \frac{1}{1+\exp(-a_k)}$

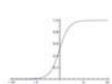
(multiclass) $y_k = \frac{\exp(a_k)}{\sum_j \exp(a_j)}$

- Putting everything together:

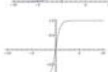
$$y_k(\mathbf{x}, \mathbf{w}) = \sigma\left(\sum_{j=0}^{M} w_{kj}^{(2)} h\left(\sum_{i=0}^{D} w_{ji}^{(1)} x_i\right)\right)$$

## Nonlinear activation functions

**Sigmoid** $\sigma(x) = 1/(1+e^{-x})$



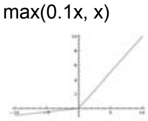
**tanh** tanh(x)



**ReLU** max(0,x)

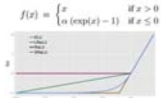


**Leaky ReLU**
max(0.1x, x)



**Maxout** $\max(w_1^T x + b_1, w_2^T x + b_2)$

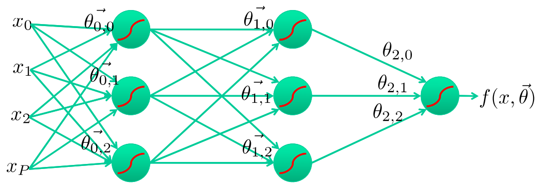
**ELU** $f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha\,(\exp(x) - 1) & \text{if } x \le 0 \end{cases}$



Andrej Karpathy

## Multilayer networks

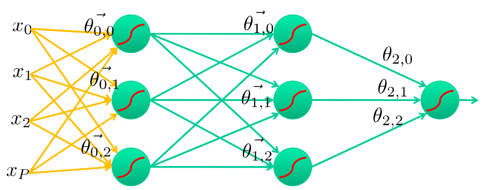
- Cascade neurons together
- Output from one layer is the input to the next
- Each layer has its own sets of weights




HKUST

## Feed-forward networks

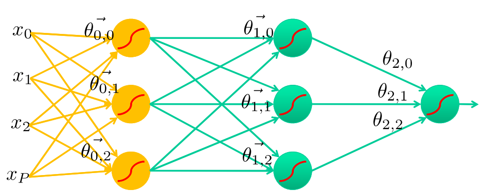
- Predictions are fed forward through the network to classify




HKUST

## Feed-forward networks

- Predictions are fed forward through the network to classify

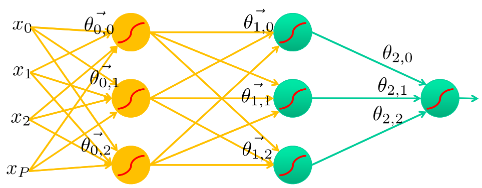



9

HKUST

## Feed-forward networks

- Predictions are fed forward through the network to classify

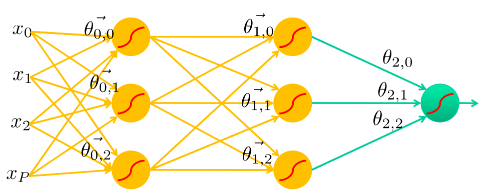



10

HKUST

## Feed-forward networks

- Predictions are fed forward through the network to classify

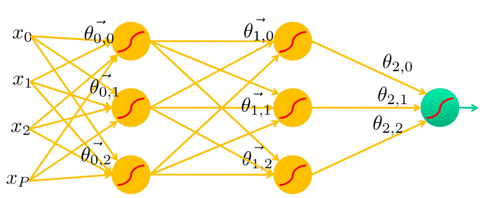



11

HKUST

## Feed-forward networks

- Predictions are fed forward through the network to classify

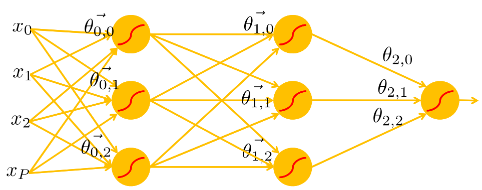



12

HKUST

## Feed-forward networks

- Predictions are fed forward through the network to classify




13

HKUST

## Deep neural networks

- Lots of hidden layers
- Depth = power (usually)

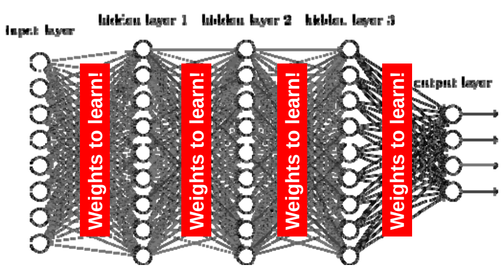



Figure from http://neuralnetworksanddeeplearning.com/chap5.html

## How do we train them?

- The goal is to iteratively find a set of weights that allow the activations/outputs to match the desired output
- For this, we will *minimize a **loss function***
- The loss function quantifies the agreement between the predicted scores and GT labels

- First, let's simplify and assume we have a single layer of weights in the network

## Classification goal


Example dataset: **CIFAR-10**
**10** labels
**50,000** training images
each image is **32x32x3**
**10,000** test images.

Andrej Karpathy

## Classification scores

$$f(x, W) = Wx \; + \mathrm{b}$$

f(x,W)

**10** numbers, indicating class scores

**[32x32x3]**
array of numbers 0...1
(3072 numbers total)

Andrej Karpathy

## Linear classifier

$$\boxed{f(x, W)} = \boxed{W}\boxed{x} \quad \boxed{+\mathrm{b}}$$

3072x1 10x1

10x1 10x3072

**10** numbers, indicating class scores

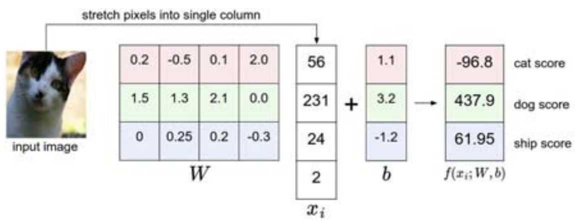
**[32x32x3]**
array of numbers 0...1

parameters, or "weights"

Andrej Karpathy

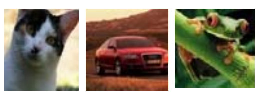
## Linear classifier

Example with an image with 4 pixels, and 3 classes (cat/dog/ship)




Andrej Karpathy

## Linear classifier



| | | | |
|---|---|---|---|
| airplane | -3.45 | -0.51 | 3.42 |
| automobile | -8.87 | **6.04** | 4.64 |
| bird | 0.09 | 5.31 | 2.65 |
| cat | **2.9** | -4.22 | 5.1 |
| deer | 4.48 | -4.19 | 2.64 |
| dog | 8.02 | 3.58 | 5.55 |
| frog | 3.78 | 4.49 | **-4.34** |
| horse | 1.06 | -4.37 | -1.5 |
| ship | -0.36 | -2.09 | -4.79 |
| truck | -0.72 | -2.93 | 6.14 |

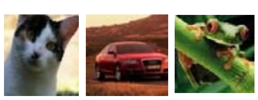
TODO:

1. Define a **loss function** that quantifies our unhappiness with the scores across the training data.
2. Come up with a way of efficiently finding the parameters that minimize the loss function. **(optimization)**

Andrej Karpathy

## Linear classifier

Suppose: 3 training examples, 3 classes.
With some W the scores $f(x, W) = Wx$ are:

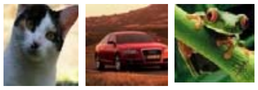


| | | | |
|---|---|---|---|
| cat | **3.2** | 1.3 | 2.2 |
| car | 5.1 | **4.9** | 2.5 |
| frog | -1.7 | 2.0 | **-3.1** |

Adapted from Andrej Karpathy

## Linear classifier: Hinge loss

Suppose: 3 training examples, 3 classes.
With some W the scores $f(x, W) = Wx$ are:



| | | | |
|---|---|---|---|
| cat | **3.2** | 1.3 | 2.2 |
| car | 5.1 | **4.9** | 2.5 |
| frog | -1.7 | 2.0 | **-3.1** |

**Hinge loss:**

Given an example $(x_i, y_i)$
where $x_i$ is the image and
where $y_i$ is the (integer) label,

and using the shorthand for the scores vector: $s = f(x_i, W)$

the loss has the form:

$$\boxed{L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)}$$

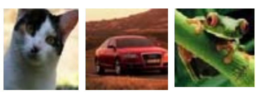
Want: $s_{y_i} >= s_j + 1$
i.e. $s_j - s_{y_i} + 1 <= 0$

If true, loss is 0
If false, loss is magnitude of violation

Adapted from Andrej Karpathy

## Linear classifier: Hinge loss

Suppose: 3 training examples, 3 classes.
With some W the scores $f(x, W) = Wx$ are:



| | | | |
|---|---|---|---|
| cat | **3.2** | 1.3 | 2.2 |
| car | 5.1 | **4.9** | 2.5 |
| frog | -1.7 | 2.0 | **-3.1** |
| Loss: | 2.9 | | |

**Hinge loss:**

Given an example $(x_i, y_i)$
where $x_i$ is the image and
where $y_i$ is the (integer) label,

and using the shorthand for the scores vector: $s = f(x_i, W)$

the loss has the form:

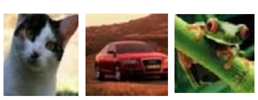
$$\boxed{L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)}$$

= max(0, 5.1 - 3.2 + 1)
+max(0, -1.7 - 3.2 + 1)
= max(0, 2.9) + max(0, -3.9)
= 2.9 + 0
= 2.9

Adapted from Andrej Karpathy

## Linear classifier: Hinge loss

Suppose: 3 training examples, 3 classes.
With some W the scores $f(x, W) = Wx$ are:



| | | | |
|---|---|---|---|
| cat | **3.2** | 1.3 | 2.2 |
| car | 5.1 | **4.9** | 2.5 |
| frog | -1.7 | 2.0 | **-3.1** |
| Loss: | 2.9 | 0 | |

**Hinge loss:**

Given an example $(x_i, y_i)$
where $x_i$ is the image and
where $y_i$ is the (integer) label,

and using the shorthand for the scores vector: $s = f(x_i, W)$

the loss has the form:

$$\boxed{L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)}$$

= max(0, 1.3 - 4.9 + 1)
+max(0, 2.0 - 4.9 + 1)
= max(0, -2.6) + max(0, -1.9)
= 0 + 0
= 0

Adapted from Andrej Karpathy

## Linear classifier: Hinge loss

Suppose: 3 training examples, 3 classes.
With some W the scores $f(x, W) = Wx$ are:

| | | | |
|---|---|---|---|
| cat | **3.2** | 1.3 | 2.2 |
| car | 5.1 | **4.9** | 2.5 |
| frog | -1.7 | 2.0 | **-3.1** |
| Loss: | 2.9 | 0 | 12.9 |

**Hinge loss:**

Given an example $(x_i, y_i)$
where $x_i$ is the image and
where $y_i$ is the (integer) label,

and using the shorthand for the scores vector: $s = f(x_i, W)$

the loss has the form:

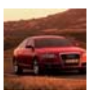
$$\boxed{L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)}$$

= max(0, 2.2 - (-3.1) + 1)
+max(0, 2.5 - (-3.1) + 1)
= max(0, 5.3 + 1)
+ max(0, 5.6 + 1)
= 6.3 + 6.6
= 12.9

Adapted from Andrej Karpathy

## Linear classifier: Hinge loss

Suppose: 3 training examples, 3 classes.
With some W the scores $f(x, W) = Wx$ are:

| | | | |
|---|---|---|---|
| cat | **3.2** | 1.3 | 2.2 |
| car | 5.1 | **4.9** | 2.5 |
| frog | -1.7 | 2.0 | **-3.1** |
| Loss: | 2.9 | 0 | 12.9 |

**Hinge loss:**

Given an example $(x_i, y_i)$
where $x_i$ is the image and
where $y_i$ is the (integer) label,

and using the shorthand for the scores vector: $s = f(x_i, W)$

the loss has the form:
$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$

and the full training loss is the mean over all examples in the training data:

$$L = \frac{1}{N} \sum_{i=1}^{N} L_i$$

L = (2.9 + 0 + 12.9)/3
= 15.8 / 3 = **5.3**

Adapted from Andrej Karpathy

## Linear classifier: Hinge loss

$$f(x, W) = Wx$$

$$L = \frac{1}{N} \sum_{i=1}^{N} \sum_{j \neq y_i} \max(0, f(x_i; W)_j - f(x_i; W)_{y_i} + 1)$$

Adapted from Andrej Karpathy

## Linear classifier: Hinge loss

**Weight Regularization**

λ = regularization strength (hyperparameter)

$$L = \frac{1}{N}\sum_{i=1}^{N}\sum_{j\neq y_i}\max(0, f(x_i;W)_j - f(x_i;W)_{y_i} + 1) + \boxed{\lambda R(W)}$$

In common use:

**L2 regularization** $R(W) = \sum_k \sum_l W_{k,l}^2$

L1 regularization $R(W) = \sum_k \sum_l |W_{k,l}|$

Dropout (will see later)

Adapted from Andrej Karpathy

## Another loss: Softmax (cross-entropy)

| | |
|---|---|
| cat | **3.2** |
| car | 5.1 |
| frog | -1.7 |

**scores = unnormalized log probabilities of the classes.**

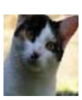
$P(Y=k|X=x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$ where $s = f(x_i;W)$

Want to maximize the log likelihood, or (for a loss function) to minimize the negative log likelihood of the correct class:

$$L_i = -\log P(Y=y_i|X=x_i)$$

Andrej Karpathy

## Another loss: Softmax (cross-entropy)



$$L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$$

| | unnormalized log probabilities | exp → | unnormalized probabilities | normalize → | probabilities |
|---|---|---|---|---|---|
| cat | **3.2** | | **24.5** | | **0.13** |
| car | 5.1 | | 164.0 | | 0.87 |
| frog | -1.7 | | 0.18 | | 0.00 |

L_i = -log(0.13) = **0.89**

Adapted from Andrej Karpathy

## How to minimize the loss function?

Andrej Karpathy

## How to minimize the loss function?

In 1-dimension, the derivative of a function:

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

In multiple dimensions, the **gradient** is the vector of (partial derivatives).

Andrej Karpathy

| | |
|---|---|
| **current W:** | **gradient dW:** |
| [0.34, | [?, |
| -1.11, | ?, |
| 0.78, | ?, |
| 0.12, | ?, |
| 0.55, | ?, |
| 2.81, | ?, |
| -3.1, | ?, |
| -1.5, | ?, |
| 0.33,…] | ?,…] |
| **loss 1.25347** | |

Andrej Karpathy

| current W: | W + h (first dim): | gradient dW: |
|---|---|---|
| [0.34, | [0.34 + **0.0001**, | [?, |
| -1.11, | -1.11, | ?, |
| 0.78, | 0.78, | ?, |
| 0.12, | 0.12, | ?, |
| 0.55, | 0.55, | ?, |
| 2.81, | 2.81, | ?, |
| -3.1, | -3.1, | ?, |
| -1.5, | -1.5, | ?, |
| 0.33,…] | 0.33,…] | ?,…] |
| **loss 1.25347** | **loss 1.25322** | |

Andrej Karpathy

| current W: | W + h (first dim): | gradient dW: |
|---|---|---|
| [0.34, | [0.34 + **0.0001**, | [**-2.5**, |
| -1.11, | -1.11, | ?, |
| 0.78, | 0.78, | ?, |
| 0.12, | 0.12, | |
| 0.55, | 0.55, | |
| 2.81, | 2.81, | |
| -3.1, | -3.1, | |
| -1.5, | -1.5, | ?, |
| 0.33,…] | 0.33,…] | ?,…] |
| **loss 1.25347** | **loss 1.25322** | |

(1.25322 - 1.25347)/0.0001
= -2.5

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

Andrej Karpathy

| current W: | W + h (second dim): | gradient dW: |
|---|---|---|
| [0.34, | [0.34, | [-2.5, |
| -1.11, | -1.11 + **0.0001**, | ?, |
| 0.78, | 0.78, | ?, |
| 0.12, | 0.12, | ?, |
| 0.55, | 0.55, | ?, |
| 2.81, | 2.81, | ?, |
| -3.1, | -3.1, | ?, |
| -1.5, | -1.5, | ?, |
| 0.33,…] | 0.33,…] | ?,…] |
| **loss 1.25347** | **loss 1.25353** | |

Andrej Karpathy

| **current W:** | **W + h** (second dim)**:** | **gradient dW:** |
|---|---|---|
| [0.34, | [0.34, | [-2.5, |
| -1.11, | -1.11 + **0.0001**, | **0.6**, |
| 0.78, | 0.78, | ?, |
| 0.12, | 0.12, | ?, |
| 0.55, | 0.55, | (1.25353 - 1.25347)/0.0001 = 0.6 |
| 2.81, | 2.81, | $\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$ |
| -3.1, | -3.1, | |
| -1.5, | -1.5, | |
| 0.33,…] | 0.33,…] | ?,…] |
| **loss 1.25347** | **loss 1.25353** | |

Andrej Karpathy

| **current W:** | **W + h** (third dim)**:** | **gradient dW:** |
|---|---|---|
| [0.34, | [0.34, | [-2.5, |
| -1.11, | -1.11, | 0.6, |
| 0.78, | 0.78 + **0.0001**, | ?, |
| 0.12, | 0.12, | ?, |
| 0.55, | 0.55, | ?, |
| 2.81, | 2.81, | ?, |
| -3.1, | -3.1, | ?, |
| -1.5, | -1.5, | ?, |
| 0.33,…] | 0.33,…] | ?,…] |
| **loss 1.25347** | **loss 1.25347** | |

Andrej Karpathy

This is silly. The loss is just a function of W:

$$L = \frac{1}{N}\sum_{i=1}^{N} L_i + \sum_k W_k^2$$

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

$$s = f(x; W) = Wx$$

want $\nabla_W L$

Andrej Karpathy

This is silly. The loss is just a function of W:

$$L = \frac{1}{N}\sum_{i=1}^{N} L_i + \sum_k W_k^2$$

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

$$s = f(x; W) = Wx$$

want $\nabla_W L$

Use Calculus!

$\nabla_W L$ = ...

Andrej Karpathy

**current W:**

[0.34,
-1.11,
0.78,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,…]
**loss 1.25347**

dW = ...
(some function
data and W)

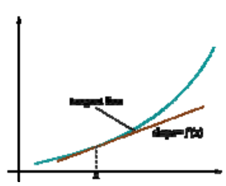
**gradient dW:**

[-2.5,
0.6,
0,
0.2,
0.7,
-0.5,
1.1,
1.3,
-2.1,…]

Andrej Karpathy

## Loss gradients

- Denoted as (diff notations): $\frac{\partial E}{\partial w_{ji}^{(1)}}$ $\nabla_W L$
- i.e. how the loss changes as a function of the weights
- We want to change the weights in such a way that makes the loss decrease as fast as possible



## Gradient descent

- We'll update weights iteratively
- Move in direction opposite to gradient:

$$\underset{\text{Time}}{\mathbf{w}^{(\tau+1)}} = \mathbf{w}^{(\tau)} - \underset{\text{Learning rate}}{\eta}\nabla E(\mathbf{w}^{(\tau)})$$

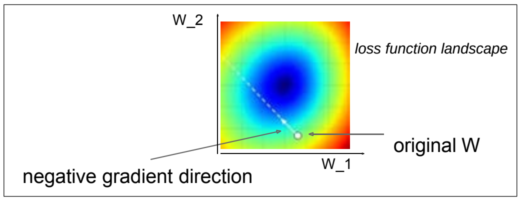



Figure from Andrej Karpathy

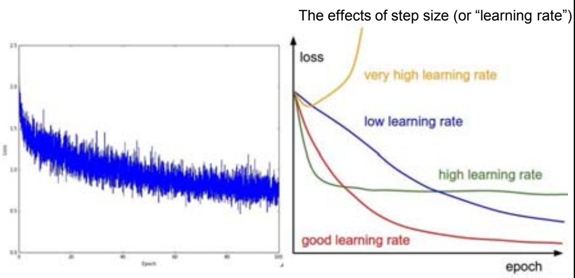
## Gradient descent

- Iteratively *subtract* the gradient with respect to the model parameters (w)
- i.e. we're moving in a direction opposite to the gradient of the loss
- i.e. we're moving towards *smaller* loss

## Mini-batch gradient descent

- In classic gradient descent, we compute the gradient from the loss for all training examples (can be slow)
- So, use only use *some* of the data for each gradient update
- We cycle through all the training examples multiple times
- Each time we've cycled through all of them once is called an 'epoch'

## Learning rate selection




Andrej Karpathy

## Gradient descent in multi-layer nets

- We'll update weights
- Move in direction opposite to gradient:

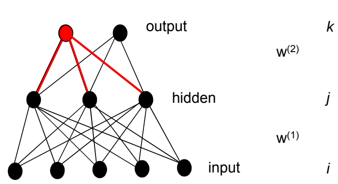
$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E(\mathbf{w}^{(\tau)})$$

- How to update the weights at all layers?
- Answer: *backpropagation* of loss from higher layers to lower layers

## Backpropagation: Graphic example

First calculate error of output units and use this to change the top layer of weights.

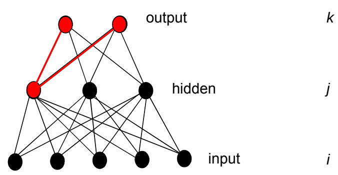
Update weights into *j*




Adapted from Ray Mooney

## Backpropagation: Graphic example

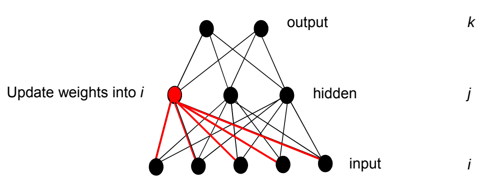
Next calculate error for hidden units based on errors on the output units it feeds into.




Adapted from Ray Mooney

## Backpropagation: Graphic example

Finally update bottom layer of weights based on errors calculated for hidden units.




Adapted from Ray Mooney

## Backpropagation

- Easier if we use *computational graphs,* especially when we have complicated functions typical in deep neural networks

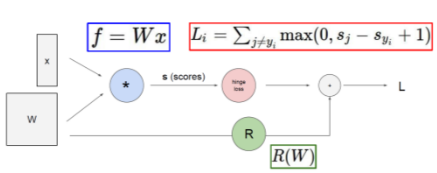
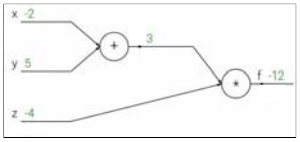



Figure from Karpathy

## Backpropagation: a simple example

$f(x, y, z) = (x + y)z$

e.g. x = -2, y = 5, z = -4

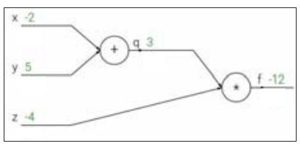



Andrej Karpathy

## Backpropagation: a simple example

$f(x, y, z) = (x + y)z$

e.g. x = -2, y = 5, z = -4




$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$

$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$

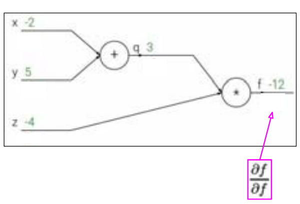
Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

Andrej Karpathy

## Backpropagation: a simple example

$f(x, y, z) = (x + y)z$

e.g. x = -2, y = 5, z = -4




$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$

$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

Andrej Karpathy

## Backpropagation: a simple example

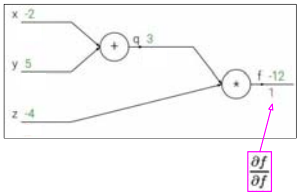
$f(x, y, z) = (x + y)z$

e.g. x = -2, y = 5, z = -4

$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$

$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$




Andrej Karpathy

## Backpropagation: a simple example

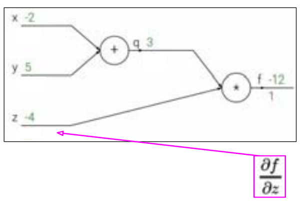
$f(x, y, z) = (x + y)z$

e.g. x = -2, y = 5, z = -4

$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$

$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$




Andrej Karpathy

## Backpropagation: a simple example

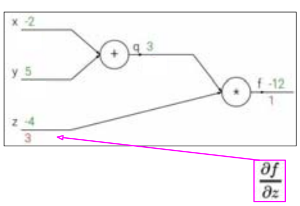
$f(x, y, z) = (x + y)z$

e.g. x = -2, y = 5, z = -4

$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$

$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$




Andrej Karpathy

## Backpropagation: a simple example

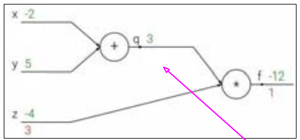
$f(x, y, z) = (x + y)z$

e.g. x = -2, y = 5, z = -4

$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$

$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$




$\boxed{\frac{\partial f}{\partial q}}$

Andrej Karpathy

## Backpropagation: a simple example

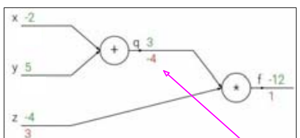
$f(x, y, z) = (x + y)z$

e.g. x = -2, y = 5, z = -4

$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$

$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$




$\boxed{\frac{\partial f}{\partial q}}$

Andrej Karpathy

## Backpropagation: a simple example

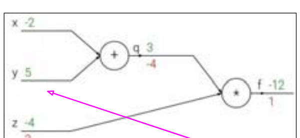
$f(x, y, z) = (x + y)z$

e.g. x = -2, y = 5, z = -4

$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$

$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$




$\boxed{\frac{\partial f}{\partial y}}$

Andrej Karpathy

## Backpropagation: a simple example

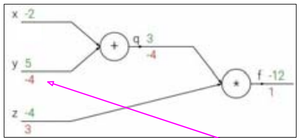
$f(x, y, z) = (x + y)z$

e.g. x = -2, y = 5, z = -4

$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$

$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$




$\frac{\partial f}{\partial y}$

Chain rule:

$$\frac{\partial f}{\partial y} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial y}$$

Upstream gradient

Local gradient

Andrej Karpathy

## Backpropagation: a simple example

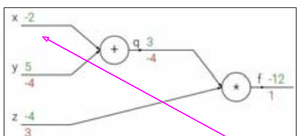
$f(x, y, z) = (x + y)z$

e.g. x = -2, y = 5, z = -4

$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$

$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$




$\frac{\partial f}{\partial x}$

Andrej Karpathy

## Backpropagation: a simple example

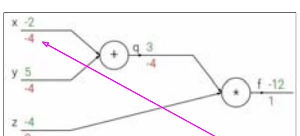
$f(x, y, z) = (x + y)z$

e.g. x = -2, y = 5, z = -4

$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$

$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

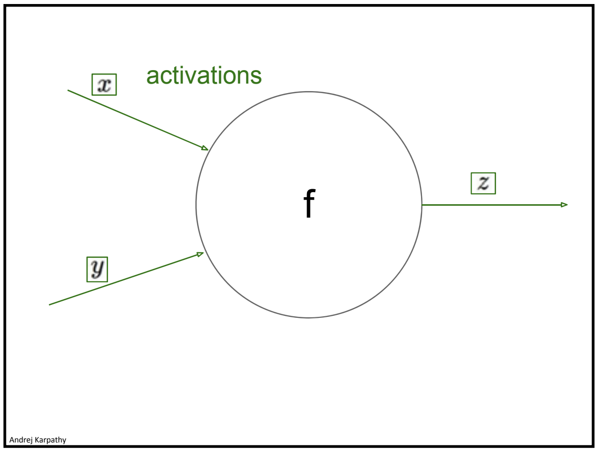



$\frac{\partial f}{\partial x}$

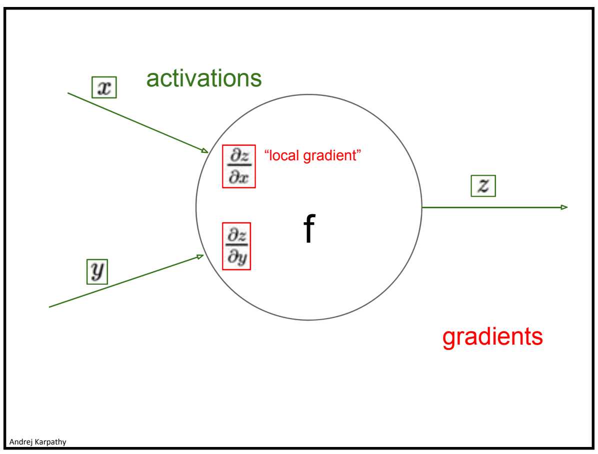
Chain rule:

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial x}$$

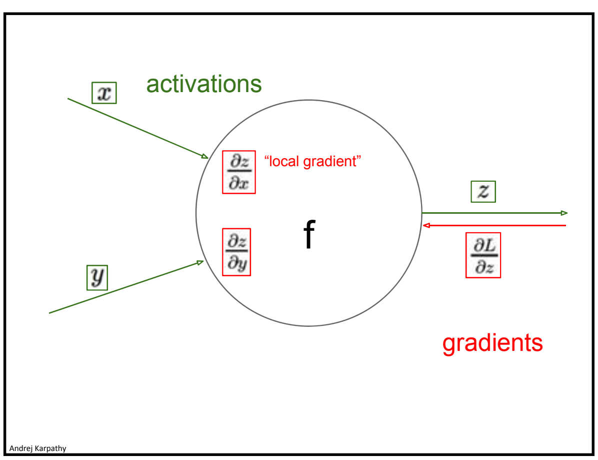
Andrej Karpathy



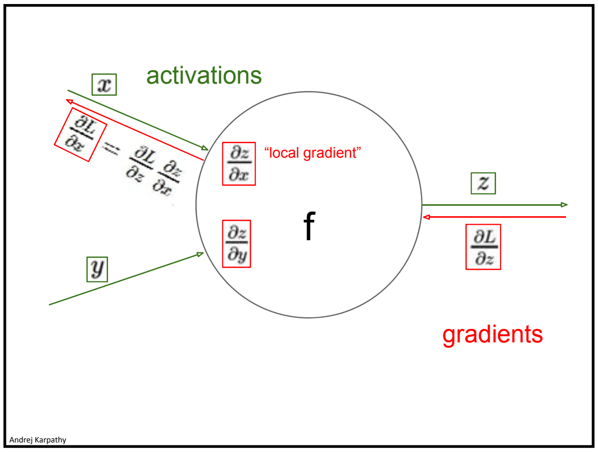
activations
$x$
$y$
f
$z$
Andrej Karpathy

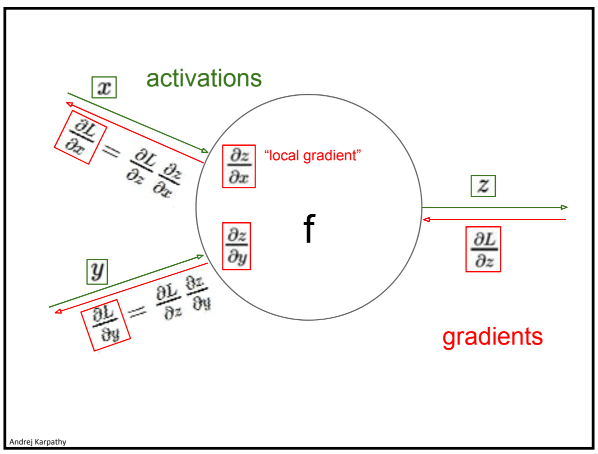



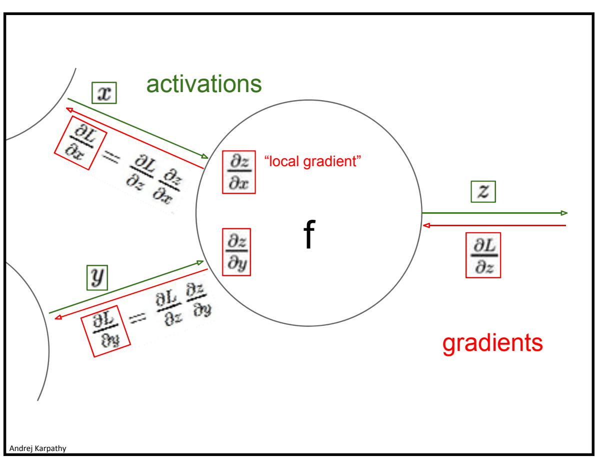
activations
$x$
$y$
$\frac{\partial z}{\partial x}$
"local gradient"
$\frac{\partial z}{\partial y}$
f
$z$
gradients
Andrej Karpathy

activations
$x$
$y$
$\frac{\partial z}{\partial x}$
"local gradient"
$\frac{\partial z}{\partial y}$
f
$z$
$\frac{\partial L}{\partial z}$
gradients
Andrej Karpathy

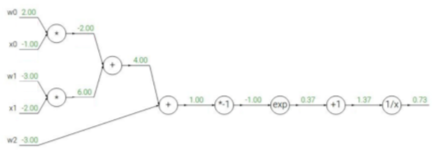

activations
"local gradient"
f
gradients
Andrej Karpathy


activations
"local gradient"
f
gradients
Andrej Karpathy


activations
"local gradient"
f
gradients
Andrej Karpathy

## Backpropagation: another example

$$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$




| | | | | | |
|---|---|---|---|---|---|
| $f(x) = e^x$ | $\rightarrow$ | $\frac{df}{dx} = e^x$ | $f(x) = \frac{1}{x}$ | $\rightarrow$ | $\frac{df}{dx} = -1/x^2$ |
| $f_a(x) = ax$ | $\rightarrow$ | $\frac{df}{dx} = a$ | $f_c(x) = c + x$ | $\rightarrow$ | $\frac{df}{dx} = 1$ |

Andrej Karpathy